Computer automated design of systems.

Vines, Larry Paul

Monterey, California. Naval Postgraduate School

COMPUTER AUTOMATED DESIGN OF SYSTEMS

Larry Paul Vines

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

Computer Automated Design of Systems

by

Larry Paul Vines

June 1976

Thesis Advisor: George J. Thaler

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Computer Automated Design of Systems		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June 1976
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Larry Paul Vines		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1976
		13. NUMBER OF PAGES 119
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Automatic control, computer automated design, optimum design feedback control, compensator, desired response,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An automated digital computer technique of control system design is presented. The emphasis is on compensator design but the method is applicable to the design of any system with free parameters. Signal representation and system response are in the time domain.

The inputs required from the engineer are the system configuration, the desired output response and the free parameters. A parameter minimization routine is then used to minimize a specific cost function and to set the free parameters. A graphical output of the desired response and actual system response is then produced for comparison by the engineer.

Computer Automated Design of Systems

by

Larry Paul Vines
Lieutenant, United States Navy
B.S.E.E., Purdue University, 1970

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the
NAVAL POSTGRADUATE SCHOOL
June 1976

ABSTRACT

An automated digital computer technique of control system design is presented. The emphasis is on compensator design but the method is applicable to the design of any system with free parameters. Signal representation and system response are in the time domain.

The inputs required from the engineer are the system configuration, the desired output response and the free parameters. A parameter minimization routine is then used to minimize a specific cost function and to set the free parameters. A graphical output of the desired response and actual system response is then produced for comparison by the engineer.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author wishes to thank Dr. George Thaler for his guidance and encouragement during the course of this investigation. Kris Butler, Ed Donnellan and M. Anderson of the W. R. Church Computer Center deserve special recognition for their patient and professional assistance. Lastly, I thank my wife, Patricia, without whom this thesis could never have been accomplished.

I. INTRODUCTION

The past two decades have been witness to an ever increasing use of the digital computer. Engineering usage and design are probably the most important justification for the large memory, high speed computers of today. Classes in computer application to engineering problems have become part of the established curricula at most universities. Numerous papers have appeared on the application of the computer to engineering problems [1] which until recently were unsolvable or at best required long, tedious procedures which gave only approximate solutions.

Control system engineering has relied increasingly on computer simulation of large scale systems to verify that design specifications have been met and to make modifications to a system even before producing a prototype. There are programs available to simulate virtually any electrical circuit, or control system, either in transfer function form or as a system of first order differential equations. Others draw the Bode, Nichols or Nyquist plots of open and closed loop systems. Some programs help design compensators which use an iterative method to achieve the desired frequency response [2] [3]. Researchers continue to better adapt the computer to engineering usage.

Cantalapiedra [4] has used an iterative method to find the optimum reduced order model for large order systems. MacNamara [5] went even further and used an iterative method to find the optimum compensation for an aircraft autopilot. It would appear that these techniques could be extended and applied to the direct simulation and design of control system circuits in the time domain.

The intent of this thesis was to develop a user oriented program which could simulate a wide variety of control systems and determine the values of the gains, poles and zeros necessary to produce a desired response. A convenient means of data card input was to be provided to specify (1) the control system which was to be optimized and (2) the desired response. A locally available function minimization routine (BOXPLX) was to be used to optimize the simulated system's output.

To simulate the system which is to be optimized, transfer functions which are commonly encountered were reduced to first order linear differential equations. These equations were then programmed so that the transfer function blocks could be connected in an arbitrary fashion by data card input. Several common nonlinear transfer blocks were also provided. The program will simulate the system with the known parameters and then allow all unknown or adjustable parameters to be fixed by the computer optimization routine to achieve the desired response.

II. PROGRAM DEVELOPMENT AND IMPLEMENTATION

A. GENERAL

Any program which is to be of maximum benifit to the user must have a simple means of data input and an output which is easy to interpret and apply to the problem at hand. The input data should have a physical significance that does not lose its relevance through the programming of numerous equations. The program should be a readily usable tool and not a problem in itself. The intent of this thesis is to present such a program, Computer Automated Design of Systems (CADS), which is readily used for simulation with optimization of the output. Optimum in the sense that the output response of a given system configuration is as near the desired response as possible.

Most control system design starts with a proposed transfer function block schematic to achieve a desired time domain response. CADS has the common transfer functions built into the program and the input data can come directly from the proposed system schematic. The transfer blocks which are available for system simulation are presented in Table I. These blocks should be adequate, either seperately or in various cascade combinations, to represent most control systems.

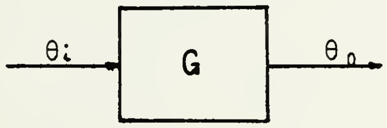
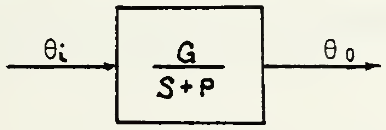
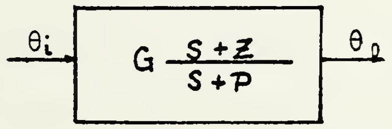
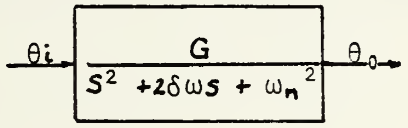
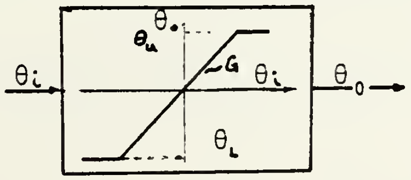
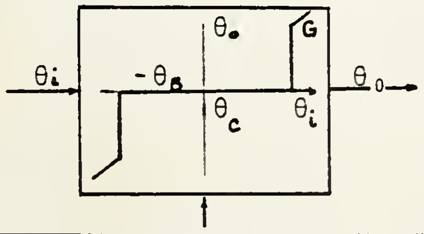
TYPE OF BLOCK	SYMBOL	EQUATION SOLVED
1		$\theta_0 = G\theta_i$
2		$\dot{\theta}_0 = -P\,\theta_0 + G\,\theta_i$
3		$\dot{\theta}_0 = -P\,\theta_0 + G(\dot{\theta}_i + Z\dot{\theta}_i)$
4		$\ddot{\theta} = -2\delta\omega_n\dot{\theta}_0 - \omega_n^2\,\theta_0 + G\theta$
5		$\theta_0 = G\theta_i \quad \theta_L \leq G\theta_0 \leq \theta_u$ $\theta_0 = \theta_u \quad \theta_u < G\theta_i$ $\theta_0 = \theta_L \quad G\theta_i < \theta_L$
6		$\theta_0 = 0 \quad \lvert\theta_c\rvert < \theta_\beta$ $\theta_0 = G\,\theta_i \quad \lvert\theta_c\rvert \geq \theta_\beta$

TABLE I Program Transfer Function Blocks

To use the program the engineer must know the system configuration in transfer block form and the desired second order output response. If other than a second order response is desired, this may be easily specified but requires some knowledge of how to input the desired response. CALS will take the transfer block system, connect it, compare the given system response to the desired response and set any free parameters to achieve the closest match possible of the system output to the desired response.

Figure 1 is representative of the type of system the program can simulate and optimize. In figure 1, the parameters of the numbered transfer function blocks are known or fixed by equipment limitations. Transfer blocks X, Y, and Z contain variable parameters which must be selected by the program to make the system output reproduce a desired output function as closely as possible.

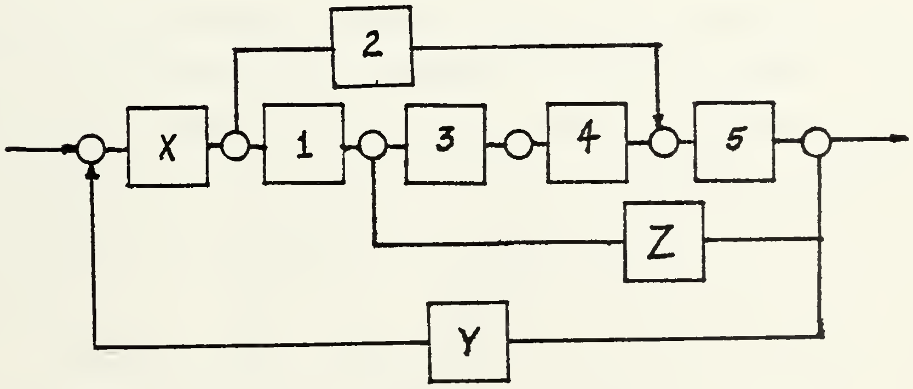


FIGURE 1 Typical System to be Optimized

B. MAIN PROGRAM

The MAIN program was developed to control the selection of the subfunctions used in the optimization process and to compute the desired response of the systems which were to be optimized. A second order step response is calculated from equation (I) as the desired response and stored in the array XDATA.

$$\theta_0 = \frac{G}{S^2 + 2\,\delta\, W_\eta S + W^2}\; u(t) \qquad \text{(I)}$$

The MAIN is essentially a bookkeeping routine which controls the program execution. The subfunction operations it controls are defined in the following sections.

C. PLANT

The common simulation routines LISA, DSL, ECAP, CSMP and INTEG were investigated in an attempt to adapt them to the general problem specified above. These programs required either the input of the system's state variable equations, Laplace transform equations or the node-transfer function pair for each system simulated. These programs are useful for simulation if all of a system's parameters have been specified. However, each simulation is problem specific and these techniques are not readily integrated with other subfunction programs. A simulation subprogram which was capable of accepting data card input to simulate a wide variety of control systems and of working with other existing subprograms had to be developed.

Subfunction PLANT was developed to simulate the systems which were to be optimized. It provides the versatility necessary to simulate numerous system configurations and is capable of working with a minimization routine to optimize the variable parameters. The program reads the transfer function block connections from data cards. This data is then used to automatically set up the system equations in state variable format. These state variable equations are solved as first order, ordinary differential equations by the Runge-Kutta-Gill forth order method. The programming was done in Fortran IV and all calculations are in double precision. The capability of connecting the transfer function blocks in any configuration (feed forward, feedback,etc.) is possible with this program. Simulation flexibility is provided by having each block within the system capable of accepting an external forcing function.

1. Block Data Card

Several possible ways of inputting the data necessary to simulate arbitrary system configurations were investigated. A means of defining a system configuration directly from a block diagram schematic was chosen as the most preferable method. Using this approach, each data card was designed to be directly related to a transfer function of the system. This resulted in having direct access to the transfer function parameters for optimization.

To simulate the system, a data card is prepared for each transfer block in the schematic diagram. The data card contains a field of numbers which specify the block number, the type of transfer function contained within the block, the input node and the output node to which the block is connected and the values of any parameters associated with the transfer function. The general format of the data cards used to input the system configuration is shown below.

```
|1        |11     |20    |40    |60        (Column Number)

BLKCCD=EEVV          G      P      Z
```

Where:

CC = Position number of the block

D = Type of block (number)

EE = Input node number

VV = Output node number

G, P, and Z are parameters of the transfer function.

2. Block Connections

The number of transfer blocks in the system and the data cards associated with each of the blocks are read upon initial entry into the simulation routine. The program then connects the transfer function blocks in the proper order by comparing the input node number of a block to the output node number of every other block. Whenever these two numbers are equal, the blocks are known to be connected and a flag is set equal to 1.0. The flags are then used to identify the input drives to each block.

3. Drives

The input quantity to each block is called the DRIVE. The program was designed to allow for multiple inputs to the system being simulated. This flexibility was achieved by making the input to each block equal to the sum of the outputs of all blocks connected to the input node plus any external forcing function (DRVIN) feeding the transfer block. The input DRIVE to a block is determined as shown in equation (II).

$$\mathrm{DRIVE}(i) = \sum \mathrm{THA}(j)*\mathrm{FLAG}(j,i) + \mathrm{DRVIN}(i) \qquad \text{(II)}$$

THA(j) is the output of block j. FLAG(j,i) is 1.0 if block j is connected to block i and zero if it is not connected. DRVIN(i) is any external forcing function specified by the user which drives block i. DRVINs must be inserted in subroutine PLANT as Fortran IV statements. The standard program has DRVIN(1) = 1.0 specified as a unit step input to the system. Problem III-E in the section Investigation of Program Performance demonstrates how multiple, time varying inputs are to be inserted in the program.

4. Standard Transfer Function Blocks

The transfer function blocks available for system simulation are shown in Table I. These blocks were selected because of their common usage in the modeling process. They were also found to be adequate either separately or in various cascade combinations to represent most control systems.

The transfer function equations for each type of block were written in state variable format and stored in an array named THA. The program reads a number from a block's data card which specifies the type of transfer function associated with the block. The system equations are then solved sequentially by selecting the type of equation associtaed with block number one, solving for its output and then sequencing to block number two, etc. The integral equations are solved by a modified RKL fourth order method in the subfunction routines. Three integration routines were necessary to store the intermediate results obtained for those equations involving a double integration. The four subfunctions, RKLDE2, RKLDE3, CCPLX and RKLDE4 are called by PLANT. RKLDE2 is used for the integration of a type two block. RKLDE3 is used for integration of a type three block and to store intermediate quantities for subfunction CCPLX. CCPLX calls RKLDE3 and RKLDE4 for integration of a type four block.

No provision has been made for the input of initial conditions or the integrators. Therefore, the integrations must start at time equal zero. The user must specify the step size to be used for integration (DT) and the problem run time (TF). To conserve computer time DT should be made as large as possible. DT = TF/1000 is suggested as appropriate in most cases. Equally important is that TF be as small as possible. Use of the program has shown that letting TF be greater than the transient response time of the system is rarely justified. For preliminary analysis TF was kept to only slightly longer than the time of the first undershoot for second order dominant systems. Because of the complexity of subroutine PLANT a flow chart of PLANT has been included as Appendix B.

5. Parameters to be Optimized

The parameters of the system which are to be optimized by the minimization routine BOXPLX must be specified in PLANT. The minimization routine returns the trial values of the variable parameters to PLANT in an array labeled 'C'. Regular Fortran IV statements equate the variable parameters to the values in this array. If a pole-zero pair were to be optimized, the following statements would be inserted in PLANT: P(i) = C(1) and Z(i) = C(2). The location for the preceding statements is clearly indicated in the program listing for PLANT.

D. DESIRED RESPONSE (XDATA)

The criteria against which the system response will be compared will vary according to the application of the system. Since most design work on control systems is done on the basis of second order dominance, the program was written to provide a second order step response with adjustable δ , W_n and gain as the basic criteria against which the simulated system will be compared. The desired δ, W_n and gain are read in as input data. The step response is then computed in the main program by subfunction RKLDEQ and stored in the array called XDATA. Any other time domain response may be specified by the user by removing the second order step response equations and replacing them with the equations of the desired response. An example is provided by problem III-E in Investigation of Program Performance. If the program is being used for simulation only and no data curve is desired, setting the input variable LEAP = 1 will cause the program to bypass the computation of the standard second order data equation.

E. COST FUNCTION (PERFORMANCE INDEX, PI)

The achieved system response is compared with the desired response and the difference is the error. The program searches for the parameter settings which will minimize this error. The cost function may be specified by the user to weight the system outputs as desired in subfunction FE. The default cost function of the program is the integral error squared. $J = \int_{0}^{TF} (Err)^2 \, dt$. An example of a weighted cost function is presented in Section III.

F. MINIMIZATION ROUTINE

1. General

The free parameters are optimized to reduce the cost function by the complex method of M. J. Box.[6] Box's constrained optimization method has been programmed at the Naval Postgraduate School by R. R. Hilleary as a subroutine called BOXPLX. This subroutine will find the minimum of an arbitrary function (cost function) subject to arbitrary explicit constraints and for implicit constraints. Explicit constraints are defined as upper and lower bounds on the free parameters. Implicit constraints may be arbitrary functions of the free parameters (e.g. $P_1 P_2 < 178$).

Two function subprograms are used to evaluate the objective function and implicit constraints, FE and KE respectively. The method BOXPLX uses to search for the values of the free parameters which minimize the cost function is explained in the computer program listing.

2. Explicit/Implicit Constraints and Start Points

BOXPLX searches a feasibility region (n-dimensional space, where n = number of free parameters) defined by upper and lower bounds on the free parameters for a minimum cost function. The smaller the region defined by these boundaries, the more rapidly the program will converge to the optimum parameter settings. Good engineering judgement will be necessary to keep the feasibility region as small as possible. The boundaries of the search region are read from data cards by the main program as the upper bound (XU) and the lower bound (XL) for each free parameter.

Implicit constraints may be any arbitrary function of the free parameter desired. If an implicit constraint such as the product of a pole-zero pair must be less than some number is to be evaluated, it must be supplied by the user to the subfunction XE. No implicit constraints were used in this thesis.

The starting values of the free parameters (XS) are read in by the MAIN from data cards. A good choice of starting values will dramatically reduce the time required for optimization. A preliminary root locus or Bode plot method of estimating the best values of the free variables should be accomplished whenever possible.

G. GRAPHICAL OUTPUT

Two subroutines were written to provide for the graphical output of the desired response and the best system response achieved by the optimization process. The user may select, by data card input, either subroutine PPLT which provides a high speed printer plot or subroutine PIC which provides a calcomp graph. Every fifth integration point stored in the arrays XDATA and THAOUT is plotted. Subroutines PPLT and PIC call the subroutines PLOTP and DRAW respectively. PLOTP and DRAW are standard plotting routines at the NPS computer facility and are not a part of the simulation program. Figure 2 diagrams the information flow and data input to the program.

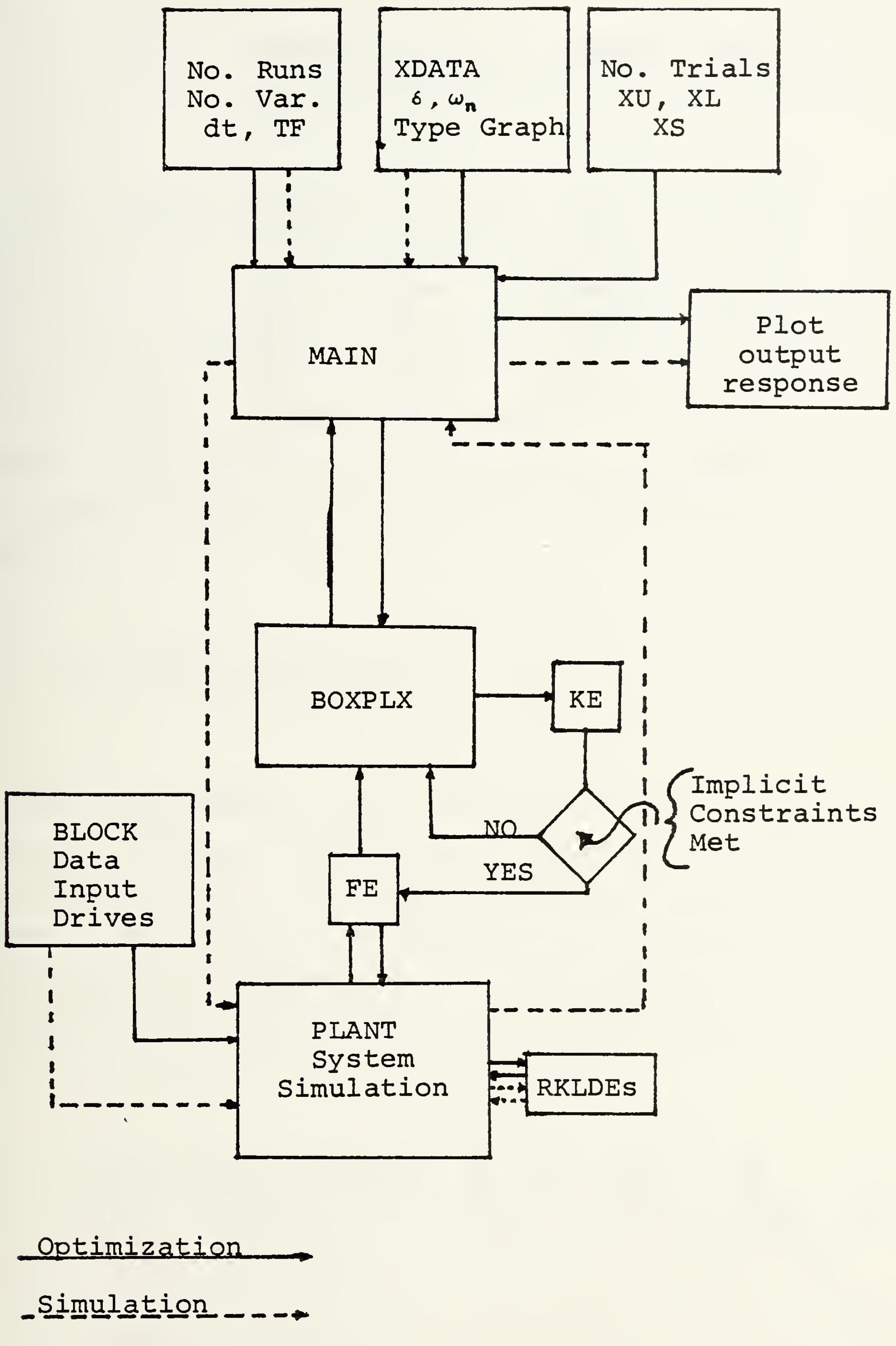


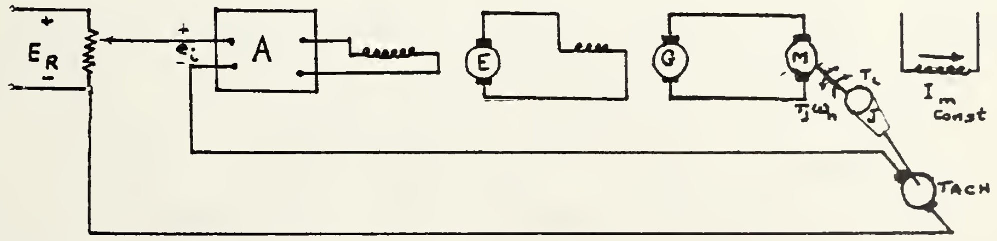
FIGURE 2 CADS Program Flow Chart

III. INVESTIGATION OF PROGRAM PERFORMANCE

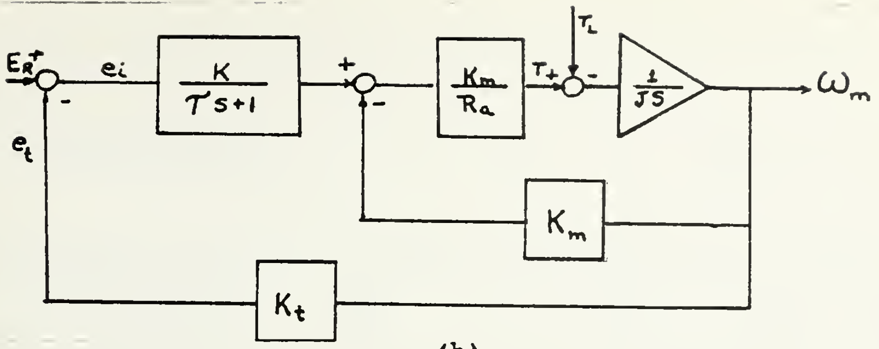
The example problems presented below were used to aid in the development of the optimization program. The order of difficulty of the problems progresses from a simple text book single variable, single input system to a multivariable operational servo drive system which has multiple inputs and discrete level feedback. An example of how a schematic diagram representation of a system is prepared for input to the program is presented prior to considering the example problems.

A. DATA INPUT, AN EXAMPLE

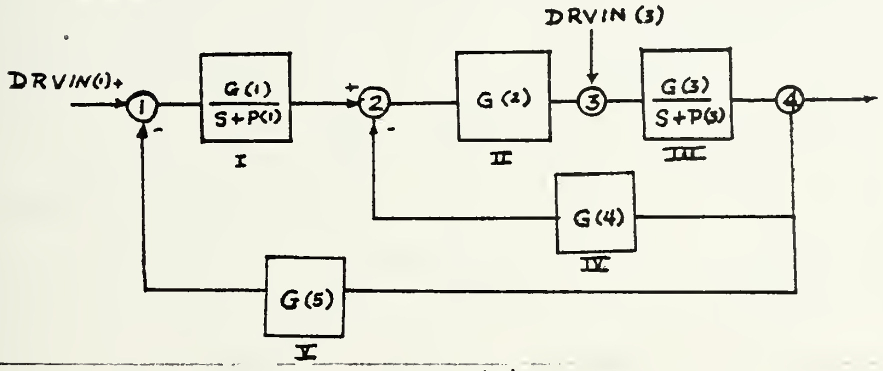
The Ward-Leonard drive system [7] shown schematically in Figure 3 (a) has two variable parameters. The gain of the amplifier and the tachometer feedback are available to adjust the system's response. To simulate the system a block diagram representation of the system is drawn as shown in Figure 3 (c) using the transfer blocks from Table I.



(a)



(b)



(c)

FIGURE 3 Ward - Leonard Speed Control System Using Feedback. (A) Schematic Diagram (B) Block Diagram (C) Block Diagram Using CADS Blocks.

Where

	$G(1) = K/\tau$	$P(1) = 1/\tau$
	$G(2) = Km/Ra$	
	$G(3) = 1/J$	$P(3) = 0$
	$G(4) = -Km$	
	$G(5) = -Kt$	
	$DRVIN(1) = Er$	
	$DRVIN\ (3) = -T_L$	

The nodes of the block diagram are then numbered sequentially 1,2,...,n. The blocks between the nodes are also numbered sequentially 1,2,...N as shown in Figure 3 (c). Data cards (N) are then prepared for each block which specify the block number, type of block, the input node, the output node, and the parameters contained within the block. In Figure 3 (c), for example, block 1 is a type two transfer block connected between nodes 1 and 2. The data card input for this block would be

```
            BLKC12=0102            K/T            1/T
```

The program reads the data card input and connects the blocks by setting a FLAG = 1. whenever the input node number to a block is the same as the output node number of any other block. The input to a transfer block is then determined to be the sum of the outputs of all blocks connected to the input node plus any external forcing function driving the input node. The program has a unit step specified for DRVIN (1). If this is the only input to the system, no action is necessary on the part of the user. If other than a unit step input to the system is desired or if there are other external forcing functions such as DRVIN (3), they must be specified and placed within the body of the subfunction PLANT as Fortran IV statements. For the example shown in Figure 3 (c), a card with the equation

$$DRVIN(3) = f(T_L)$$

would have to be inserted preceding the drive equations. An example of how multiple, time varying drives are specified is given in section III. E.

When optimizing a system, some of the input quantities will be unknown or variables. These variables must also be assigned within subroutine PLANT. To optimize the variables K_1 and K_t of Figure 3 the following two statements would be inserted in PLANT:

```
G(1) = C(1)
G(5) = C(2)
```

where C(1) and C(2) are the variables which will be optimized by subroutine BOXPLX.

B. TACHOMETER FEEDBACK

The first optimization problem attempted was one which had an exact solution that can be found by algebraic methods. An instrument servo [8] with unity feedback and forward transfer function

$$G(S) = \frac{1000}{S(S+10)} \qquad \text{(III)}$$

was to be compensated with tachometer feedback as shown in Figure 4. The only specification for the system's performance of this single variable, second order system was the simple requirement that the closed loop roots have a $\delta = 0.7$.

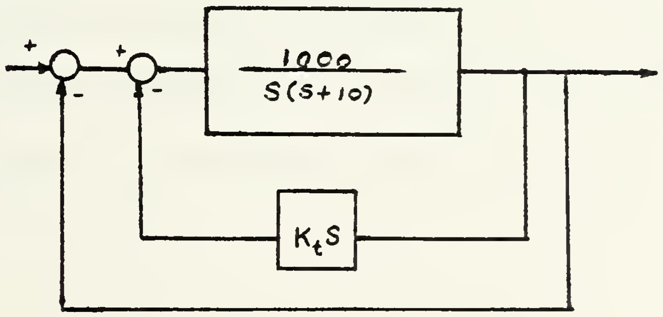


FIGURE 4 Tachometer Compensated System

The system shown in Figure 4 was redrawn as Figure 5 in order to achieve the tachometer feedback. The characteristic equation of the system shown in figure 5 is

$$S^2 + (10 + 10^3 K_t)S + 10^3 = 0 \qquad \text{(IV)}$$

The required $K_t = 0.0343$ may be calculated from equation IV.

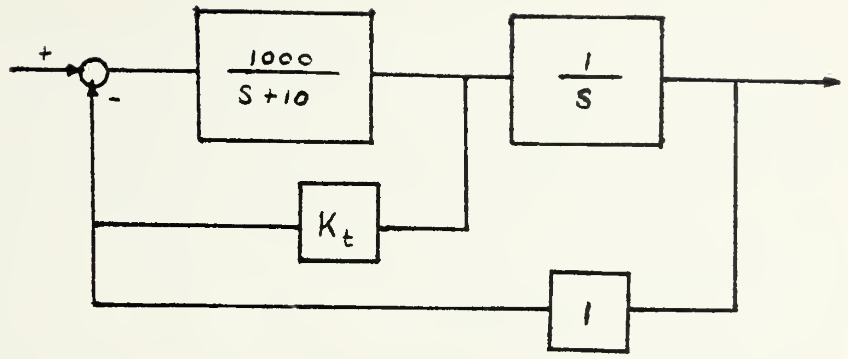


FIGURE 5 CADS Block Diagram of Tachometer System

CADS was programmed to optimize the system with the variable, K_t, specified to be between the limits $0.01 < K_t < 1.0$. The desired response was specified to be the standard second order step response for $\delta = 0.7$, $Wn = 1000$. CADS determined the optimum value for K_t to be $K_t = 0.0343$. Figure 6 shows the system's step response and the desired response are virtually superimposed.

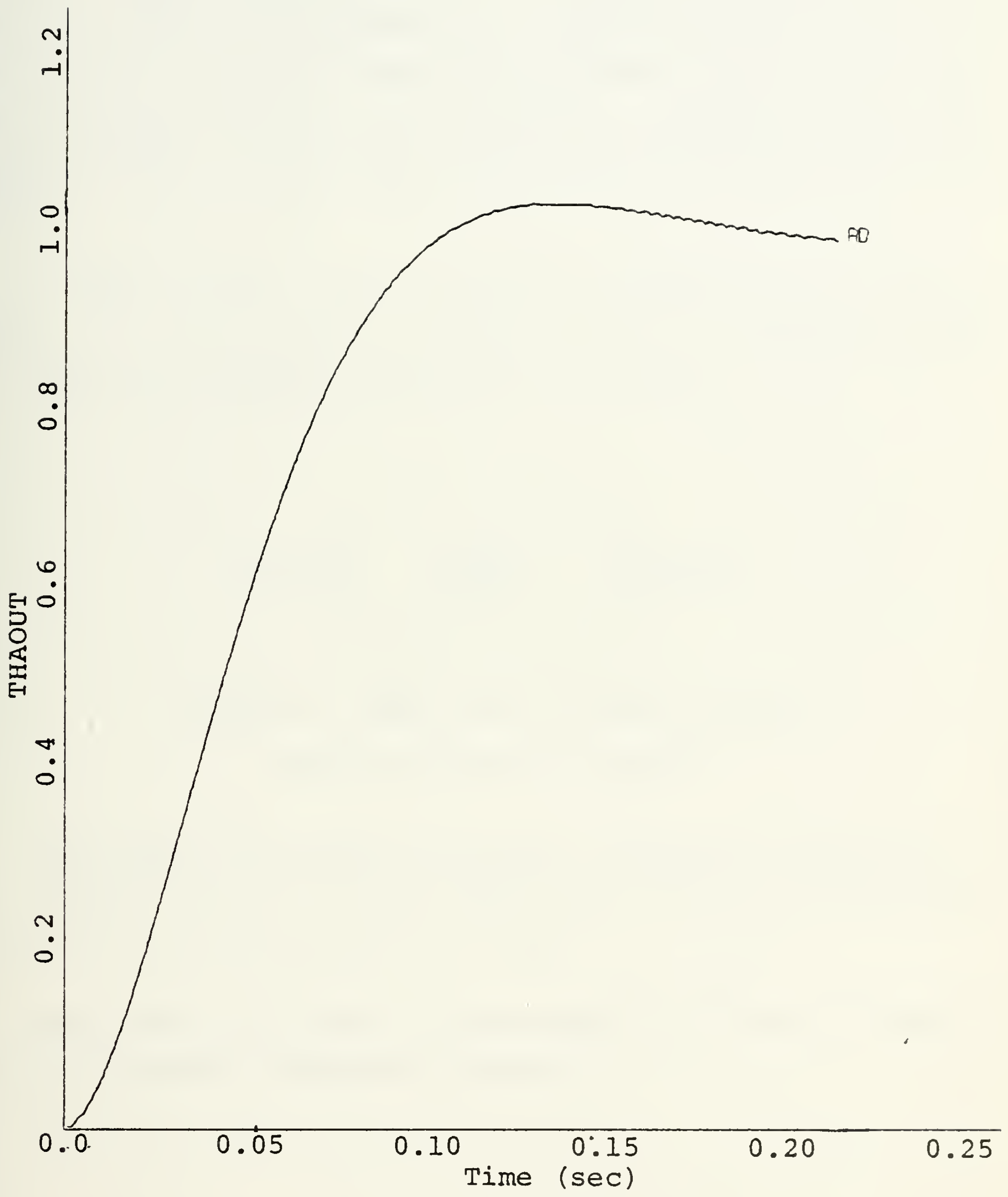


FIGURE 6 CADS Compensated Tachometer Feedback System Response

The type one system, in the preceding example, allowed derivative feedback from the forward path without requiring a block which was capable of differentiation. One may encounter a type zero system or a system which cannot be configured to provide derivative feedback from the forward path. The possibility of providing a pseudo derivative feedback using a type three block was investigated for these cases.

A type three block with Z = 0 and G = P is shown in Figure 7. If the pole is placed far out on the real axis, this block approximates derivative feedback.

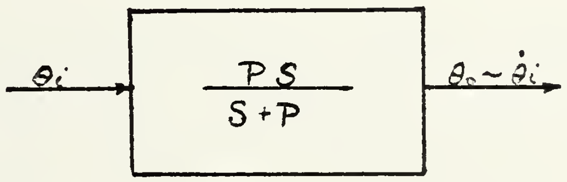


FIGURE 7 Type Three Block used for Pseudo Tachometer Feedback

The effect of using this pseudo tachometer feedback on the complex roots of the closed loop system is negligible. It does add a real root at P = -964.

The system of Figure 5 was redrawn as shown in Figure 8 using the pseudo tachometer feedback.

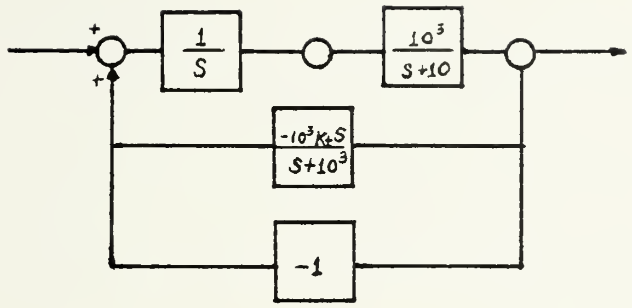


FIGURE 8 Pseudo Tachometer Feedback

The optimization program calculated K_t = 0.0352 for the above configuration. This gave a δ = 0.715.

The system response and the desired response are shown in figure 9. The two responses are again nearly superimposed. This method of providing derivative feedback has the disadvantage of adding an integration step to the problem solution with the concomitant increase in problem solution time.

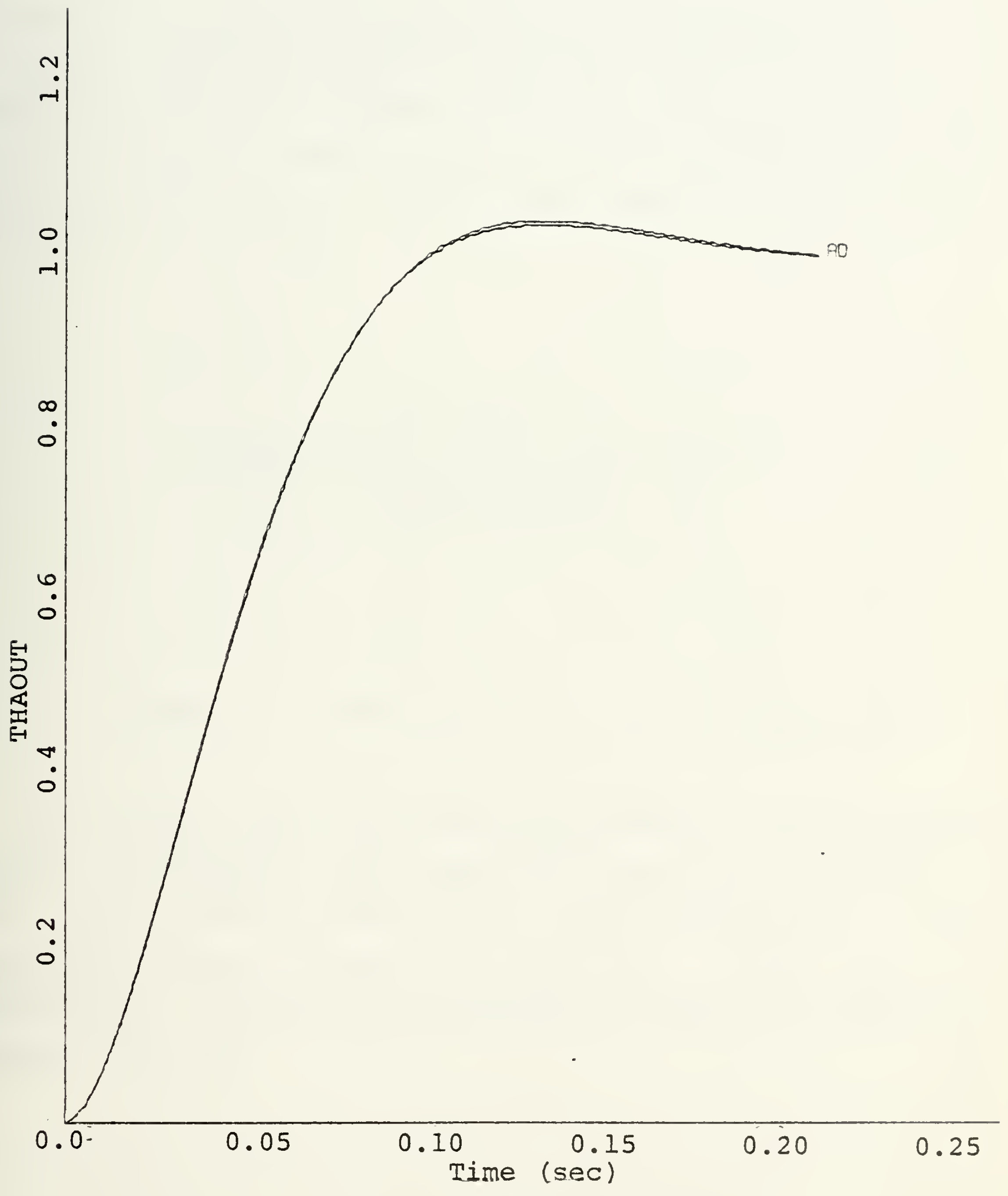


FIGURE 9 Pseudo Tachometer Compensated System's Response

C. CASCADE COMPENSATION

Having proven the feasibility of the program optimizing a single variable system where an exact solution was available, the next problem considered extending the problem scope to a third order plant with two variables. The unstable plant that was to be compensated is shown in Figure 10.

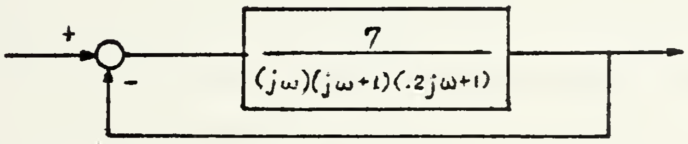


FIGURE 10 Third Order Plant to be Compensated

The plant was to be stabilized using a single section cascade compensator. The compensated plant was required to have $M_{pw} < 2$, without reducing the error coefficient. [8] The compensated system is shown in Figure 11. To keep the error coefficient constant, the compensator used was a simple lag network ($\tau_1 < \tau_2$).

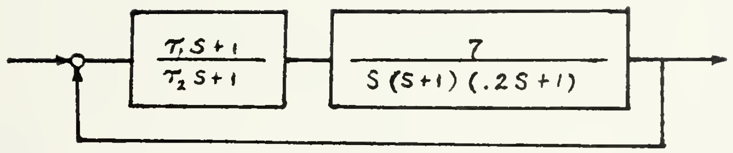


FIGURE 11 Lag Compensated System

A compensator which would meet the required specifications was calculated by conventional methods as a check on the program's performance. The Bode plot of the uncompensated and conventionally compensated system is shown in Figure 12. The values of τ_1 and τ_2 for Figure 11 which would meet the design requirements were determined to be τ_1 = 10., τ_2 = 100. The compensated system's response using these values for the compensator is shown in Figure 13.

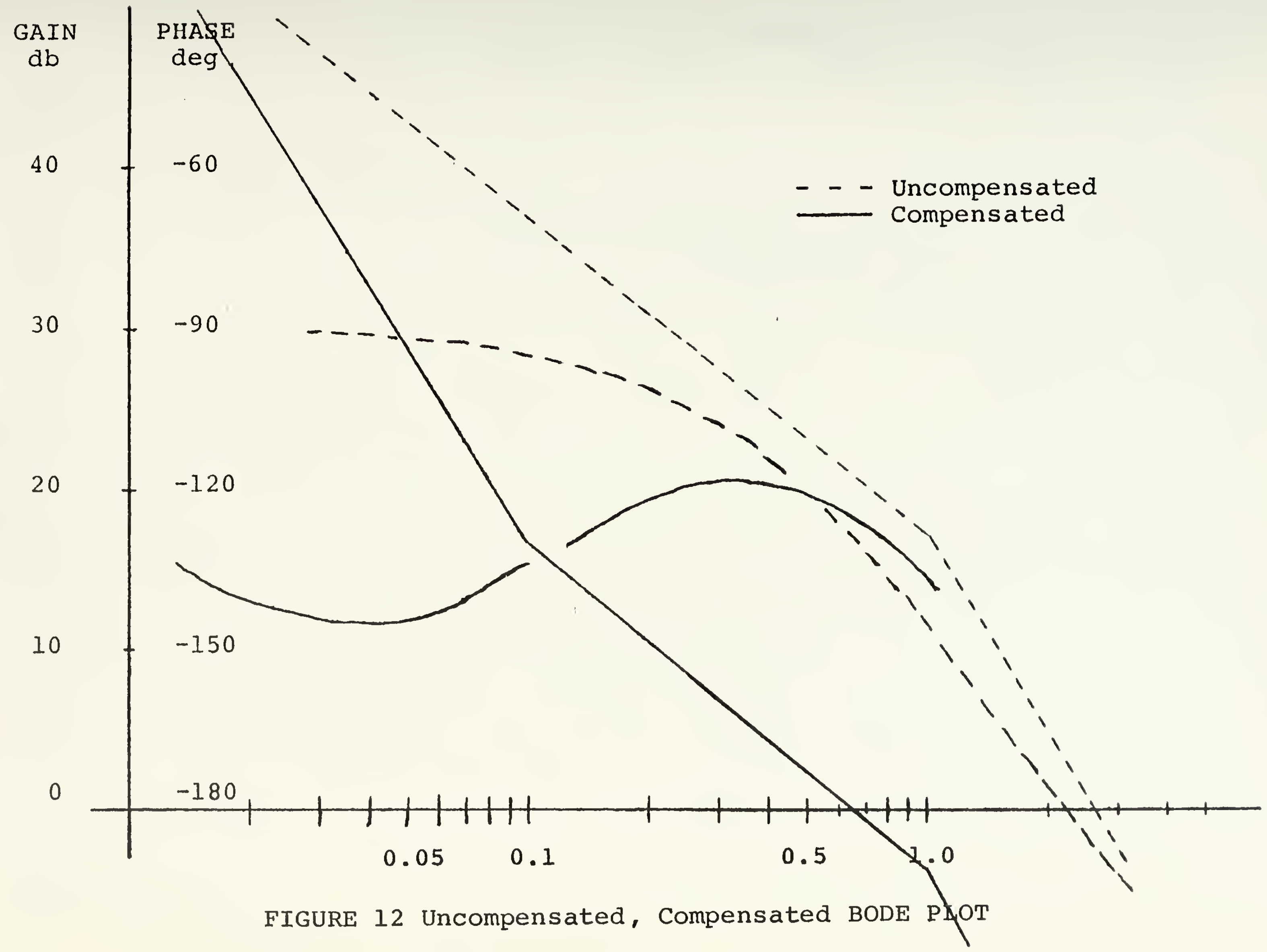


FIGURE 12 Uncompensated, Compensated BODE PLOT

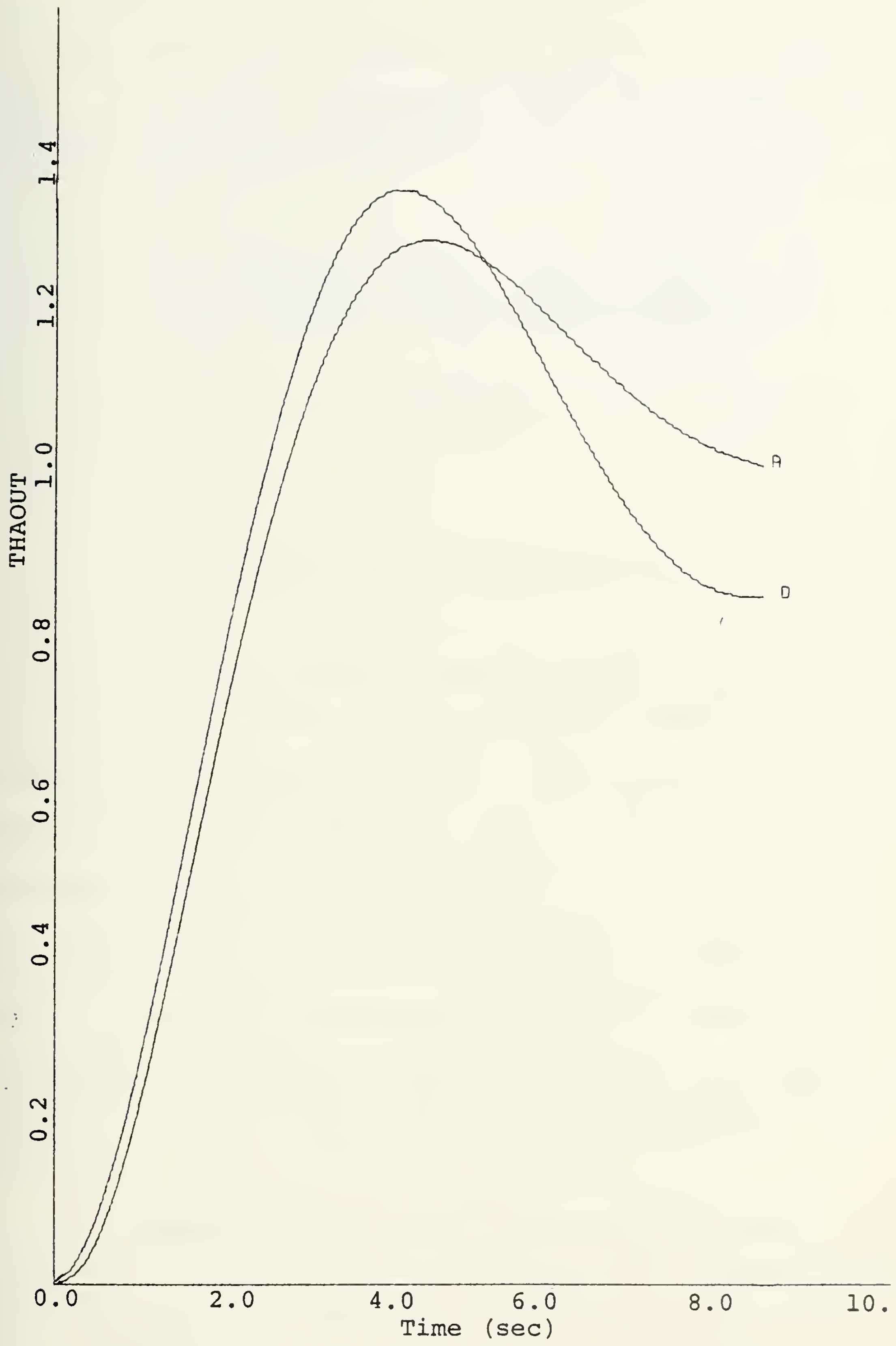


FIGURE 13 Conventionally Compensated System Response

The compensated system was redrawn as shown in Figure 14 using the standard program transfer blocks available for system simulation.

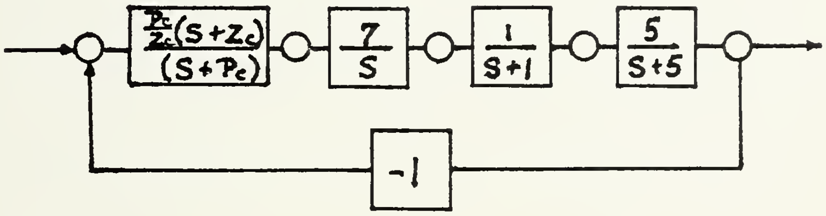


FIGURE 14 CADS Lag Compensated System

A standard second order response of $\delta = .3$, $W_n = 0.75$ was chosen as the desired response for the optimization program to match. The limits placed on the optimization program were $.001 < P_C < .1$ and $.01 < Z_C < 1$. Arbitrary values to begin optimization were specified as $P_{Co} = .01$ and $Z_{Co} = .1$ in the logarithmic centers of the search zones. CADS determined $\tau_1 = 8.078$ and $\tau_2 = 67.47$ as the optimum parameter settings. Figure 15 shows the desired response and the program compensated system response.

The response of the system compensated by CADS is more nearly the desired response than is the conventionally compensated system. One should remember that the specified response is for a true second order system whereas, the compensated system is forth order. Therefore, a perfect match cf desired versus actual responses could nct be ebtained.

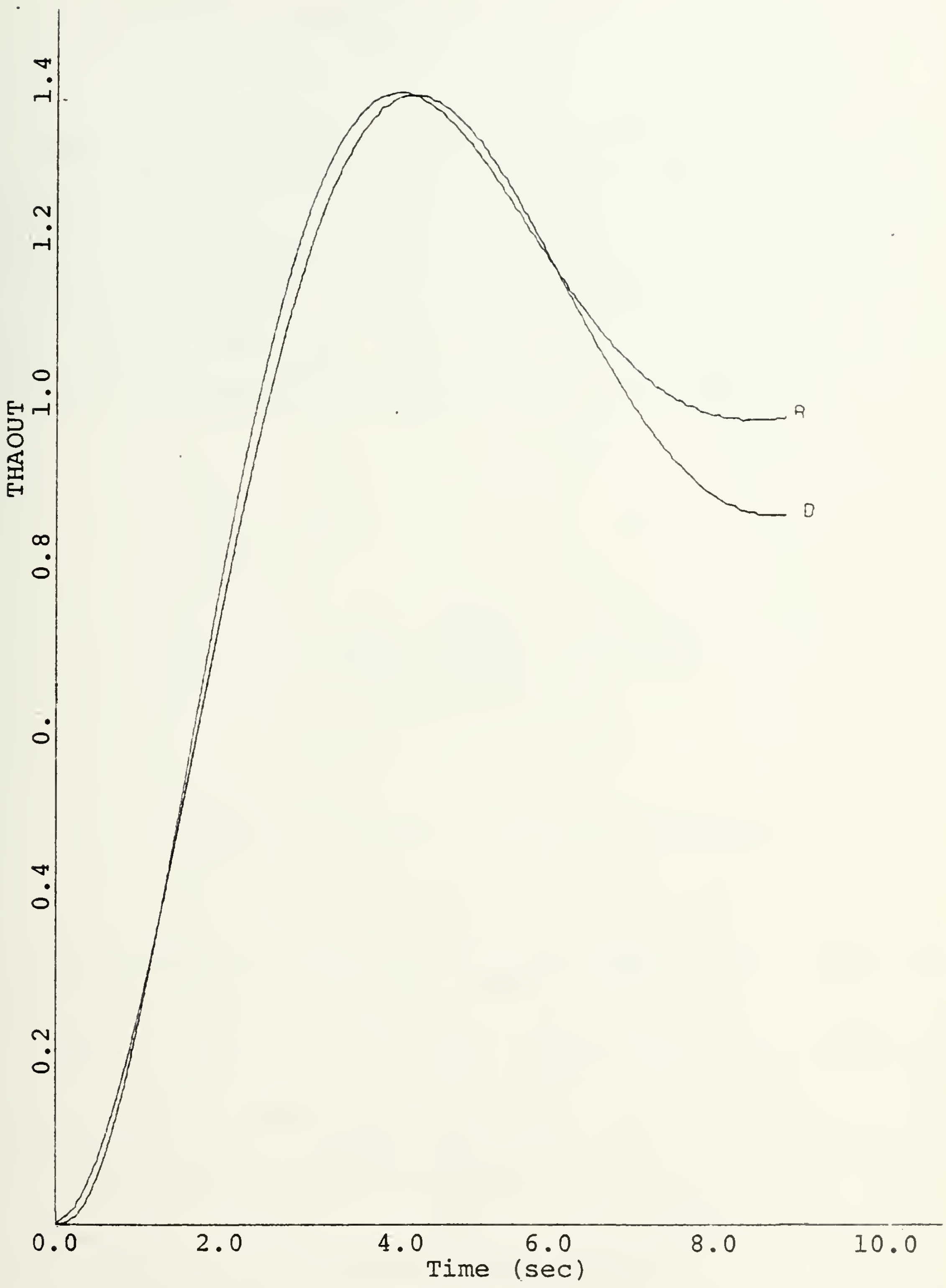


FIGURE 15 CADS Compensated System Response

D. CASCADE LEAD COMPENSATION

The complexity of the next problem to be solved by CADS was extended to five free parameters. The plant shown in figure 16 was to be used to follow a unit amplitude sine-wave input of 200 rad/sec. The output amplitude was to be almost exactly the same as the input amplitude, and the output could not lag the input by more than 10°. Two sections of cascade compensation were to be used. [8]

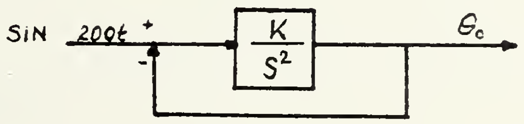


FIGURE 16 Plant for Lead Compensation

The specified requirements may be interpreted as an open loop gain > 15 db and a phase angle $\sim$ 90° at $W_n = 200$ from a Nichols plot. A cut and try solution on a Bode plot showed that a gain of 3 X 10^6 and two phase lead compensators with a double zero at Z = 70 and a double pole at P = 700 will satisfy the closed loop magnitude and phase requirements. Figure 17 shows the system response and desired response obtained for these values. The magnitude of the compensated system's response is 93% of the desired response and lags by 8.12°.

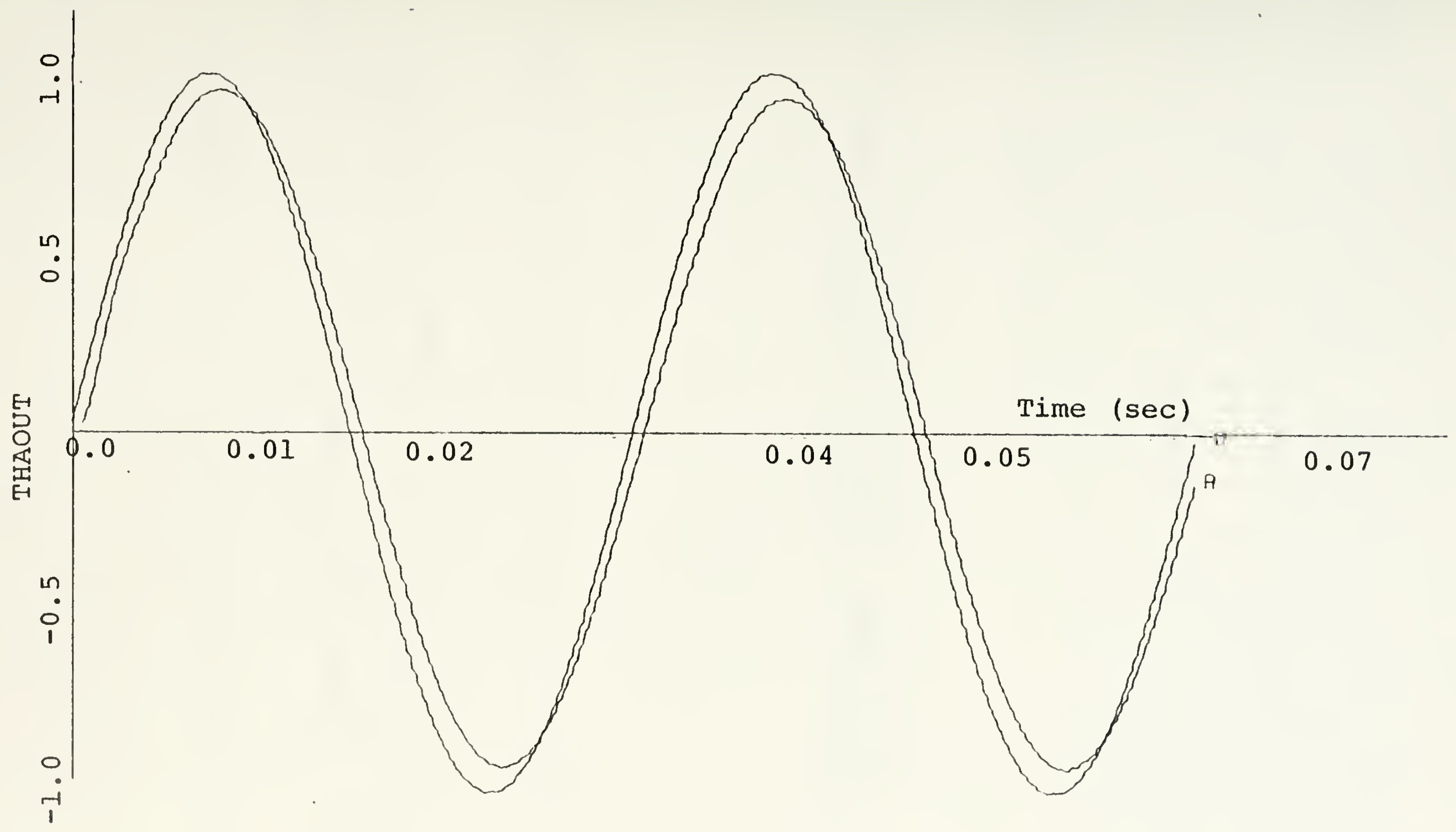


FIGURE 17 Conventional Lead Compensated System Response

The problem was then run on the optimization program with the block connections as shown in Figure 18. The free parameters were the poles and zeros of the compensators and the gain of the plant. The desired data curve of XDATA = sin(200t) was generated from the standard second order step response by setting $\delta = 0$, $W_n = 200$ and using X(2) as the desired response. The initial search zone limits were specified as $600 < P_i < 800$, $60 < Z_i < 80$, $2.8 \times 10^6 < G(3) < 3.2 \times 10^6$.

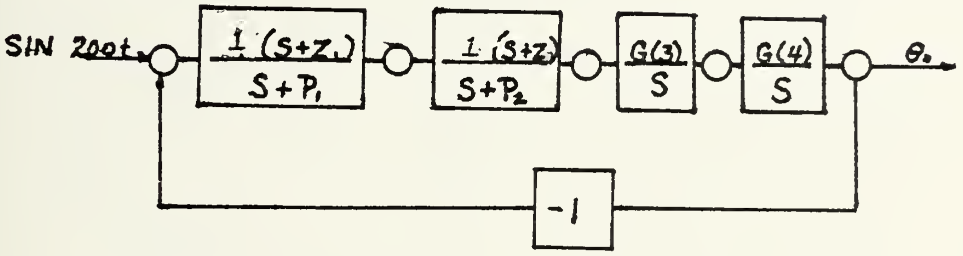


FIGURE 18 CADS Block Diagram for Lead Compensation

The optimization program solution went to the lower limits for both poles and the upper limits for both the zeros and the gain. The limits of the search zones were relaxed to $500 < P_i < 600$, $80 < Z_i < 90$ and $3.2 \times 10^6 < G(3) < 3.4 \times 10^6$. The program again placed the free variables on the limits of $P_i = 500$, $Z_i = 90$, and $G(3) = 3.4 \times 10^6$. The system and desired response for these values is shown in Figure 19. The system magnitude and phase are much closer to the desired output response than the response of the conventionally designed compensator.

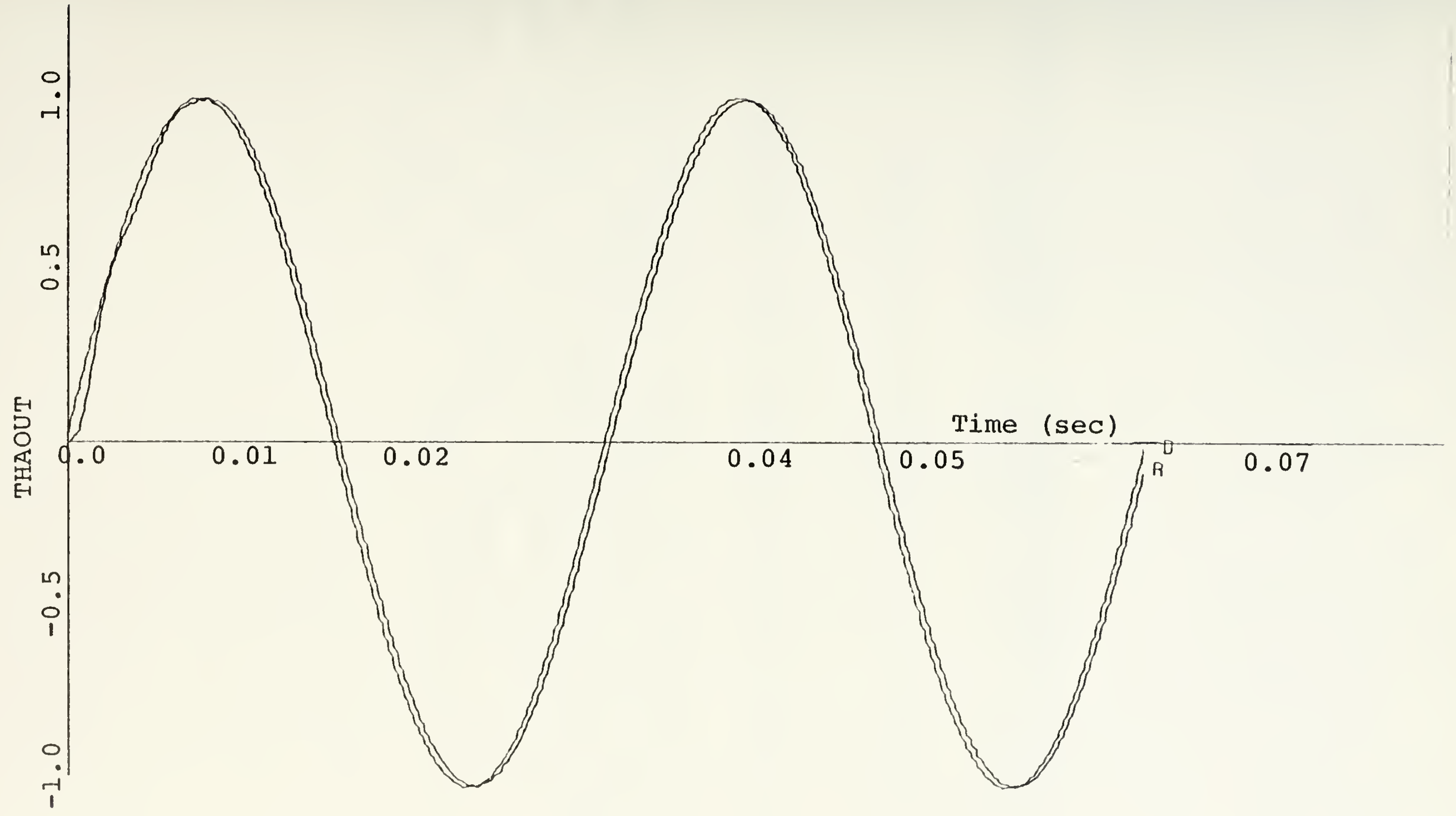


FIGURE 19 CADS Lead Compensated System Response

The phase difference is only 4.33°. Continued relaxation of the boundaries produced the compensated system shown in Figure 20. Figure 21 is a plot of the system's response for these values. The system's response is improved in that it lags the input by only 3.46° and the magnitude is essentially the same as the input magnitude.

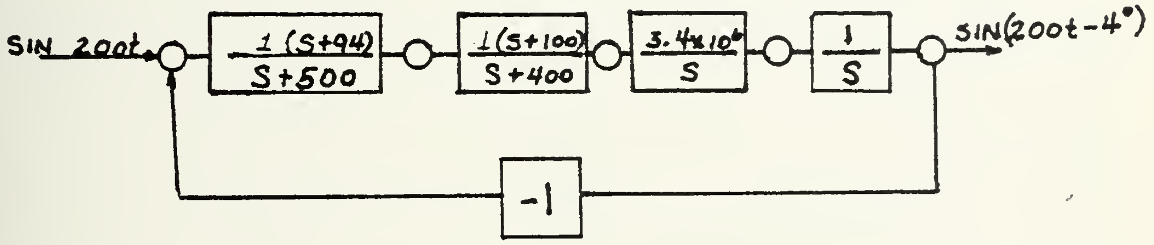


FIGURE 20 CADS Compensated System Block Diagram

The optimization program was activated at T = 0 although the problem specifications were for the steady state response. The problem was rerun with the optimization process started after the transient had died out. The same values for the optimum compensator were obtained. Apparently the small initial transient did not effect the problem solution.

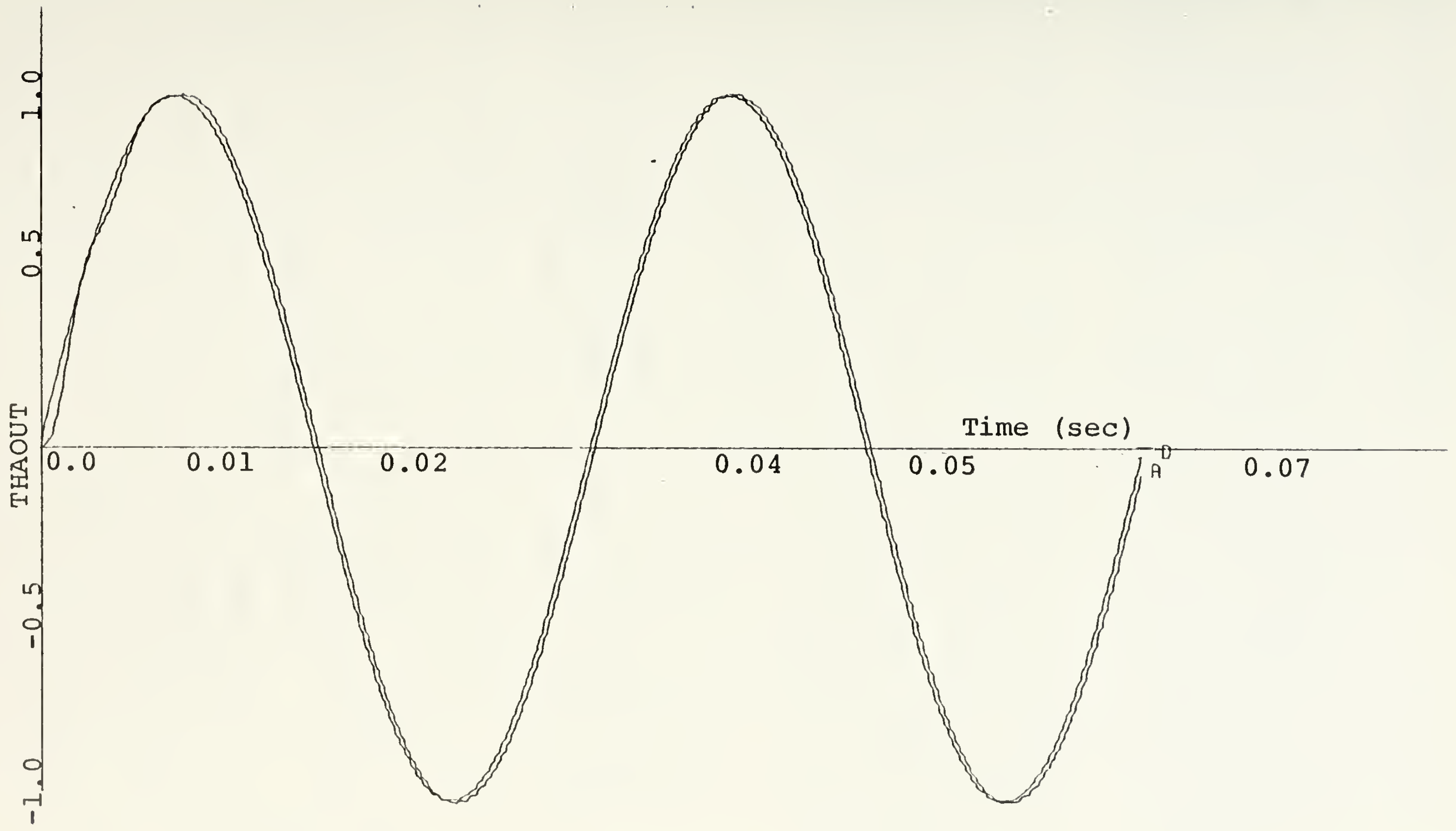


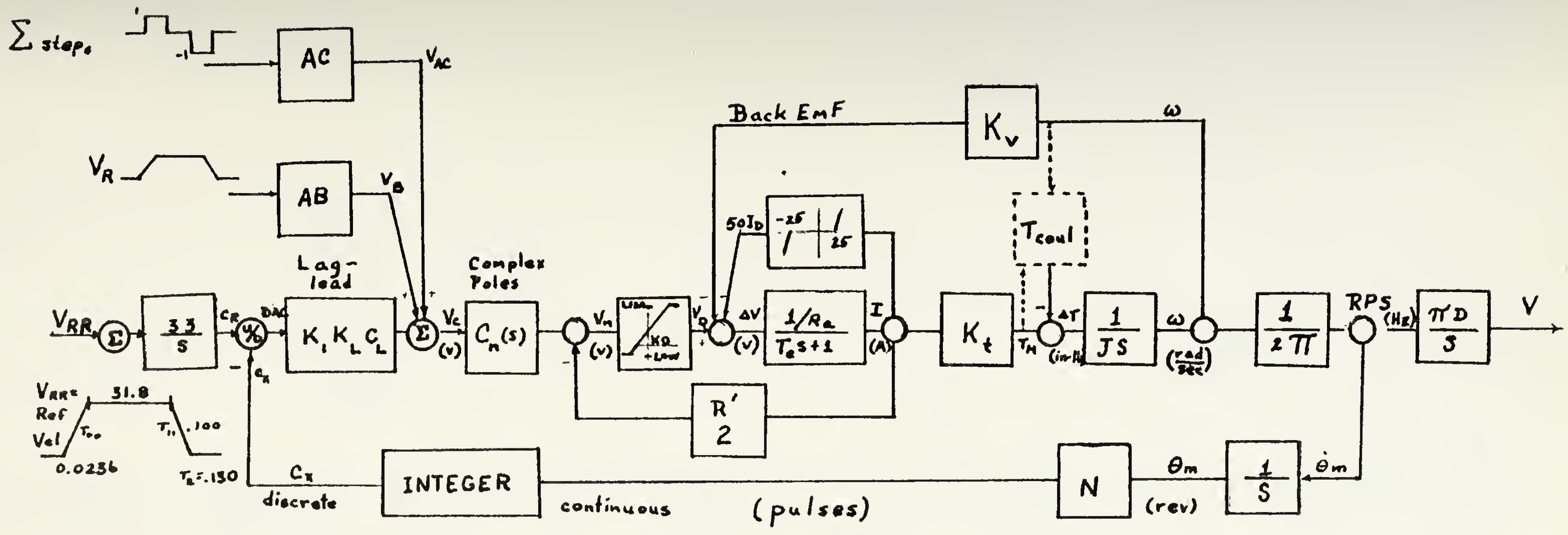
FIGURE 21 CADS Relaxed Boundary System Response

E. SERVO SYSTEM COMPENSATION

The foregoing examples of the simulation - optimization program were simple examples which could obviously be quickly solved with the standard cut and try methods. A more complex and challenging example of program usage is presented by the system shown in Figure 22.

The system is an operational servo drive mechanism with multiple inputs and discrete level feedback. This highly nonlinear system's output was to follow the input as closely as possible and in steady state ($t \geq 75$ms) there was to be very little noise ripple. The free parameters of the system are the poles and zeros of the compensator, C_L, and the poles of the noise suppressor C_n.

To simulate and optimize the system it was redrawn using the available program blocks as shown in Figure 23. During simulation it was found that the current limiter for I_D was not needed because $|I_D| < 25$ A and the limiter was removed to decrease program run time. The desired response (XDATA) was written as a set of three equations. These equations were then used to replace the second order step response equations in the standard program. DRVIN(1), (4) and (5) were also written as a set of equations and placed in subroutine PLANT. The discrete level feedback to block two was achieved by making DRVIN(2) = INTGER(THA(12)). These changes to the main program and PLANT were all that were neccessary to simulate this system. The implementation of these changes to the program is shown on pages 80 and 81.



$AC = \dfrac{A J R_a}{K_t K_D}$

$AB = \dfrac{K_v}{r K_D}$

$A = \frac{1}{2} \dfrac{V_p^2}{X_a}$

$X_a = \frac{3}{8}$ in.

$V_p = 31.8$ (in/sec)

$V_R = A * \Sigma$ Ramps

$C_L(s) = \dfrac{(1+S/50)^2}{(1+S/20)(1+S/500)}$

$C_n(s) = \dfrac{\omega_n^2}{S^2 + 2\delta\omega_n S + \omega_n^2}$

$\delta = 0.5$, $\omega_n = 1300$

$T_c = 2.425$ (lb.)

$Te = \dfrac{L}{R_a}$

$K_1 = 1.6$

$K_L = 3.48$

$K_D = 2$

Lim = 20

Low = -4

N = 52

$K_v = 0.086$

$K_T = 0.765 \left(\frac{in.lb.}{A}\right)$

$R_a = 0.5$ (Ω)

$L = 50 (\mu H)$

$J = 0.0016$ (in.lb. sec^2)

$r = \dfrac{(\pi D)/8}{2\pi}$

$D = 4.016$ in.

$r_o = \dfrac{\pi D}{8}$

FIGURE 22 Servo Drive Mechanism, Original Compensation

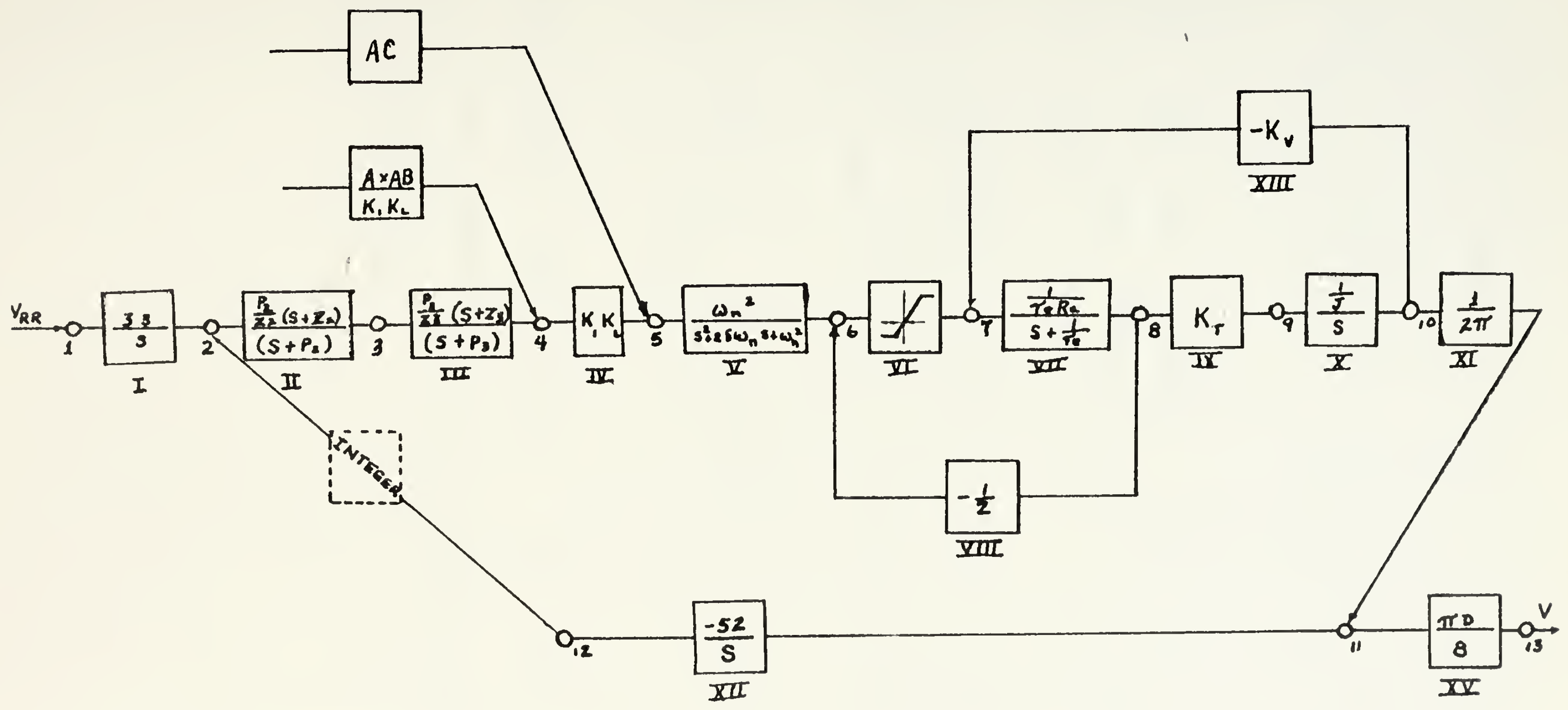


FIGURE 23 CADS Dagram of Servo Drive Mechanism

The system optimization was initially broken into two parts. This was to reduce the number of variables that were to be optimized per run and to obtain near optimum starting values for the free parameters. The above steps were taken in an effort to decrease the computer time required for a solution. The first run was to optimize the compensator, C_I, independent of the noise suppressor. The second optimization run was to select values for the noise suppressor, C_N. The rationale behind this separation was that the two circuits perform different functions and should therefore be initially separable in their effects on the system. The final optimization run was to be made with all free parameters available to the program for optimization. The search zone centered on the values found above.

Figure 24 shows the response for the system as originally compensated. The optimization run to set the values for C_L resulted in $Z_1 = 43.5$, $P_1 = 21.0$, $Z_2 = 47.5$, $P_2 = 592.0$. Figure 25 shows the simulation of the system using these values. The initial velocity overshoot has been reduced and the average velocity after the transient appears more equally distributed above and below the desired velocity.

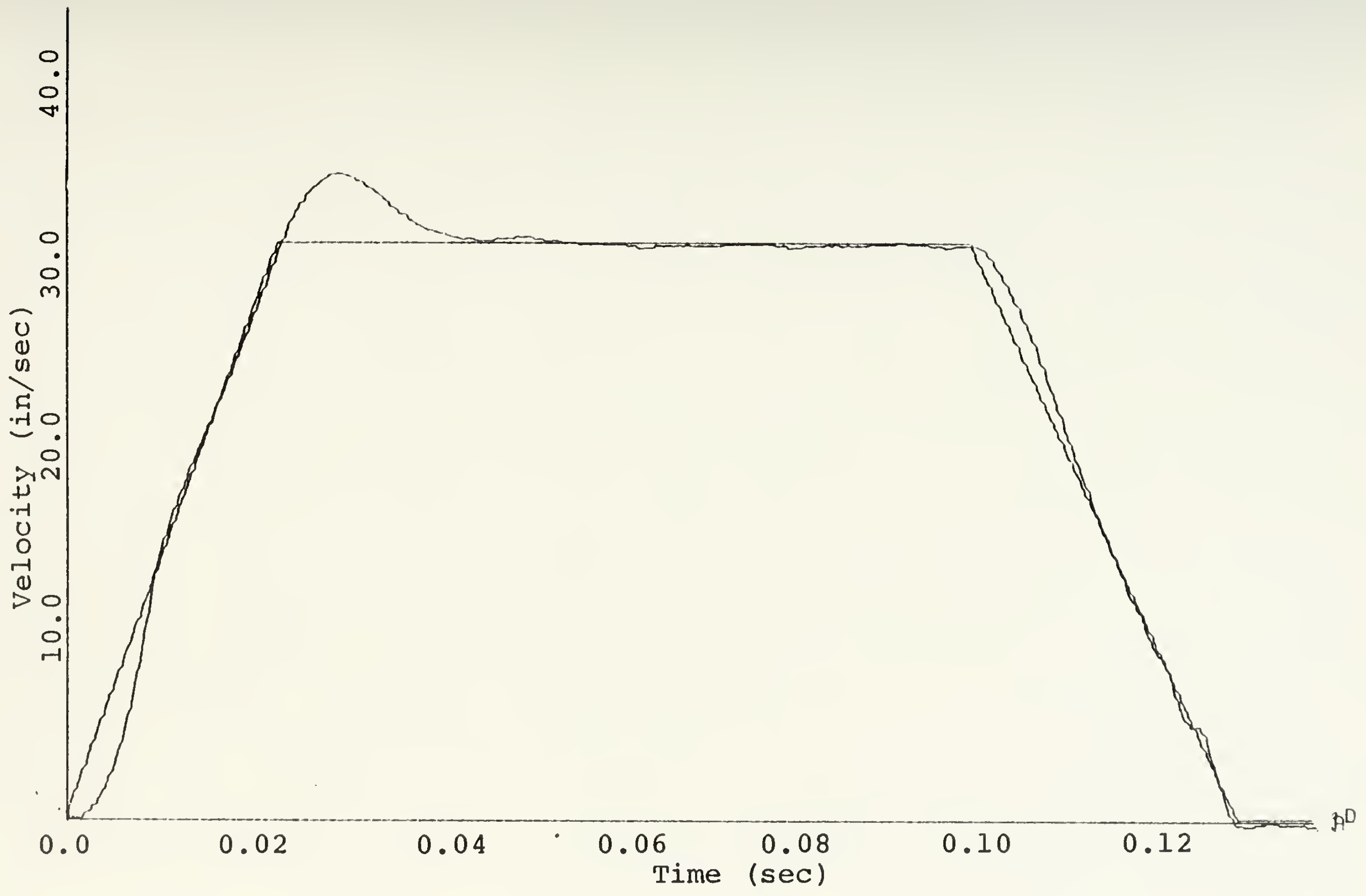


FIGURE 24 Original Servo System Response

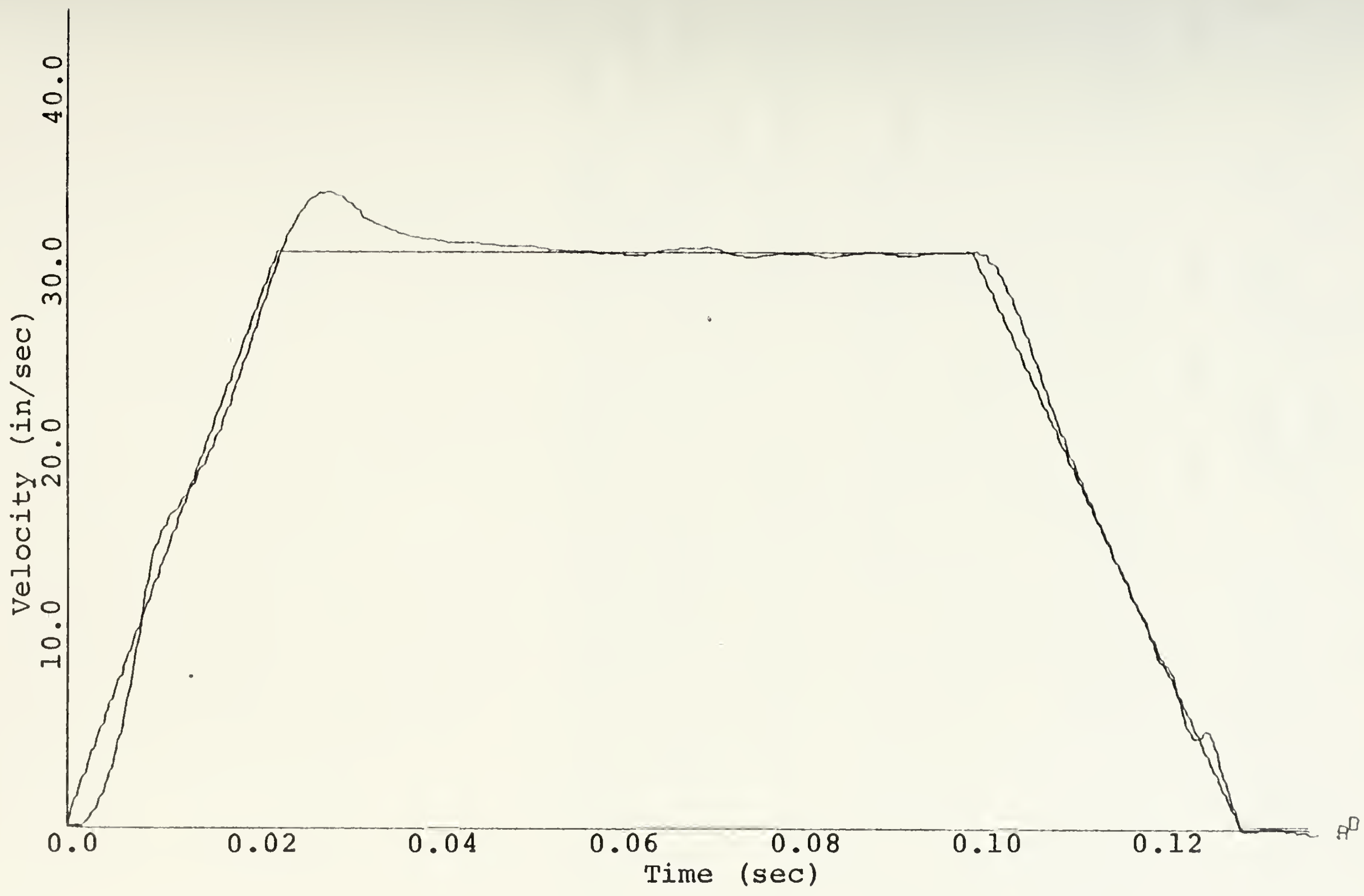


FIGURE 25 CADS Compensated C_L System Response

The initial optimization run for setting the values of the noise suppressor used the same cost function which had been used for all previous optimization runs $J = \int Err^2\,dt$. This resulted in $\delta = 0.2$ (lower bound), and $W_n = 1500$ (upper bound). Figure 26 shows the result of using this cost function was to further decrease the overshoot. However, it allowed larger switching or noise transients as a result of weighting large transient errors more than the lesser noise jitter. To overcome this problem, the cost function was changed to $J = \int |Err|*t*dt$. This was to weight the steady state errors more heavily than the transient errors. Using this cost function the values set by the optimization program were $\delta = .2$ and $W_n = 1430$. The damping factor, δ, was again placed on the lower limit. The initial overshoot was still improved over the original system's response but the switching transients were not improved over the previous optimization trial. The results of the simulation run using these values is shown in Figure 27.

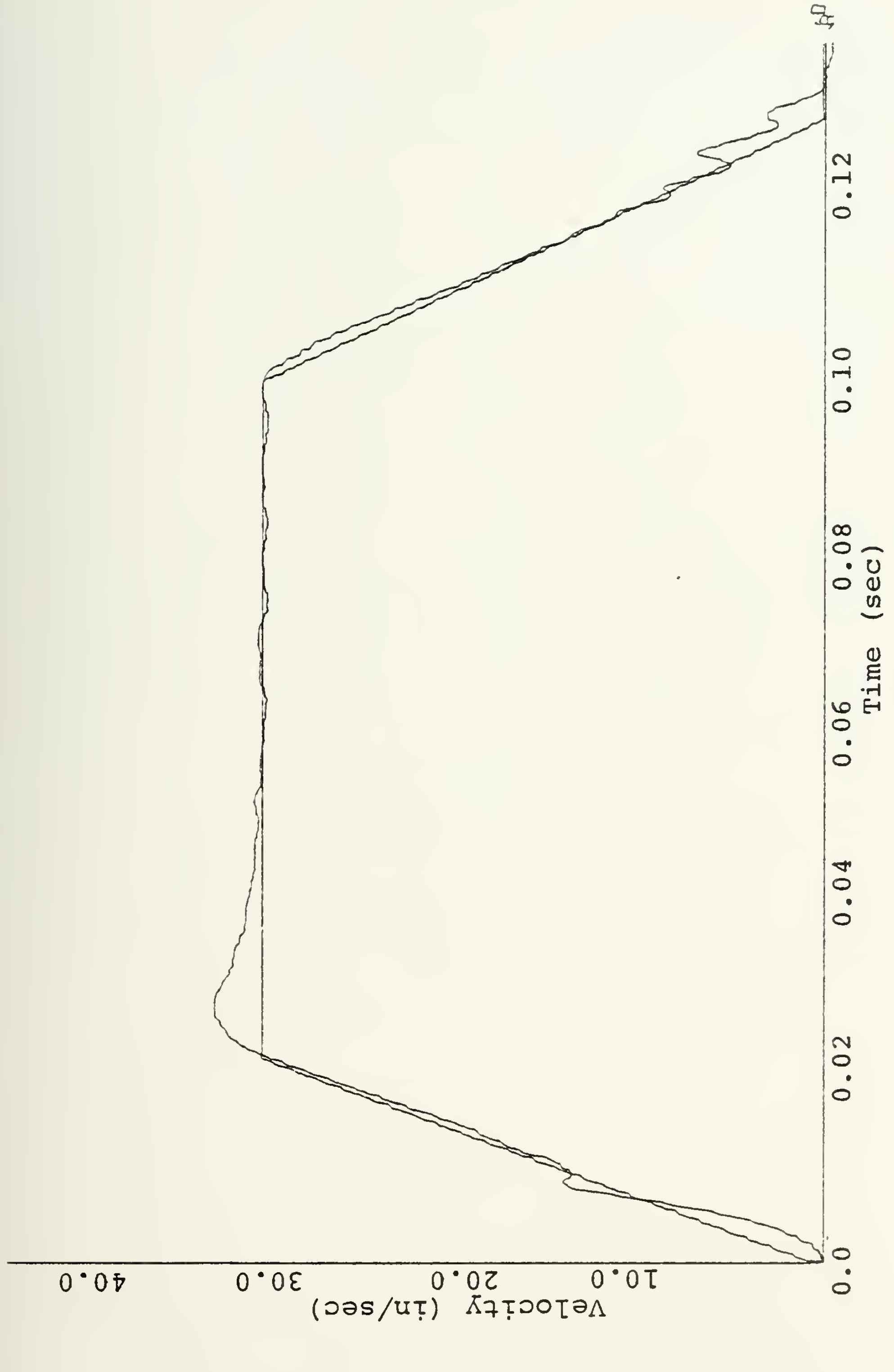


FIGURE 26 CADS Compensated C_N-1 System Response

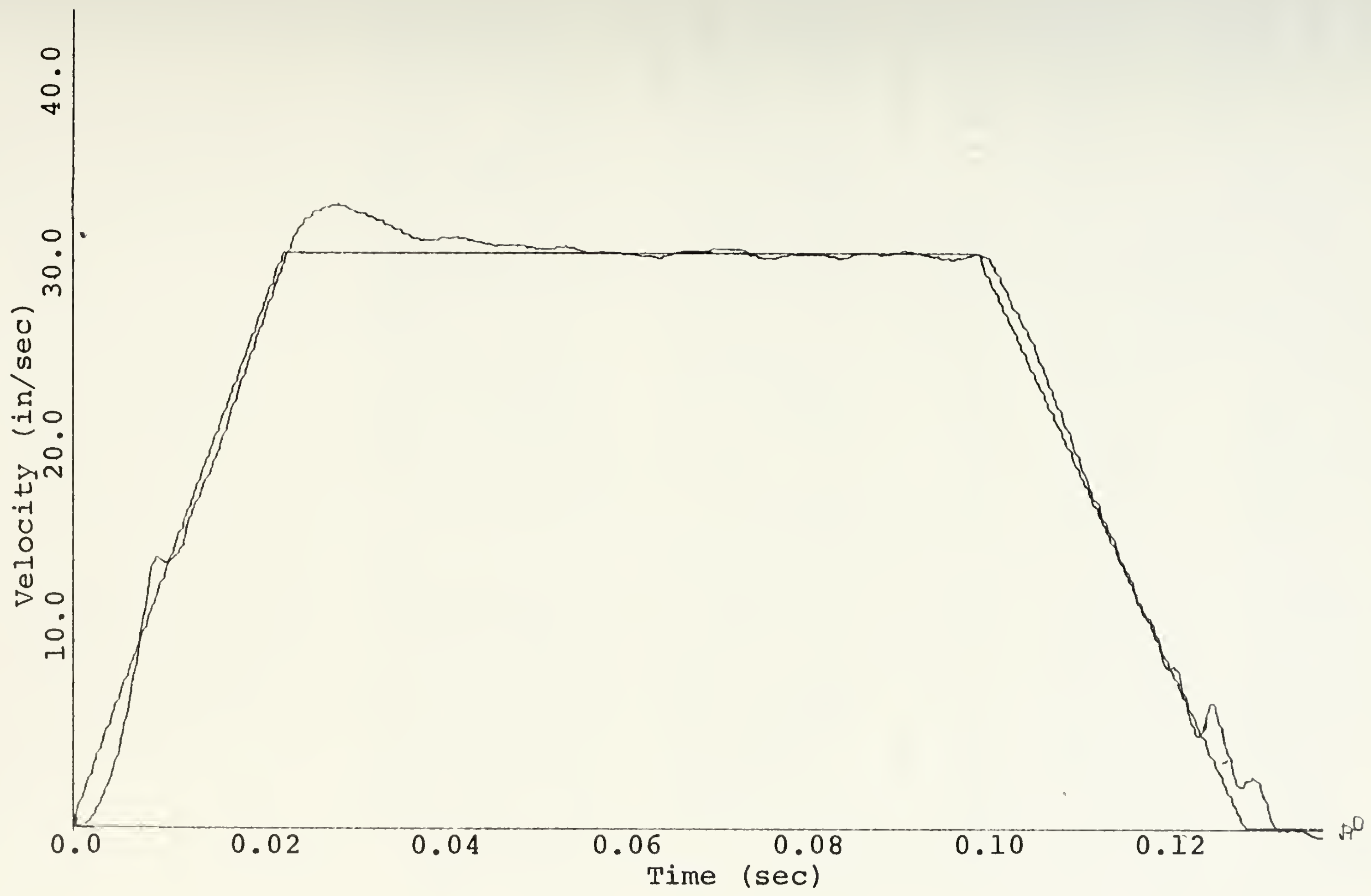


FIGURE 27 CADS Compensated C_N-2 System Response

A final optimization run was made with all six variables free. The performance index was changed to the weighted cost function shown in equation V. The weighting factor was computed to equally weight the start up transient and the steady state responses in an effort to reduce the switching transients.

$$J = \int |Err|\,dt \qquad 0.0 \le T \le 0.045$$
$$J = \int 4.14 * |Err|\,dt \qquad 0.045 < t \le TF \qquad (V)$$

The optimum compensator values established by CADS were $P_2 = 27.1$, $Z_2 = 52.5$, $P_3 = 521.$, $Z_3 = 48.1$, $\delta = 0.32$ and $W_n = 1268$. Figure 28 shows the fully compensated system's response. The initial overshoot has been reduced and the average velocity in steady state is closer to the desired value than the original system's response. However, the transient error on shut off is still present. Table II. summarizes the results of the optimization of the servo system.

The CPU time required for the optimization process was not decreased by splitting the problem into two separate parts. The time required for each optimization run on the individual compensators was the same as the time required to optimize the complete system with six variables.

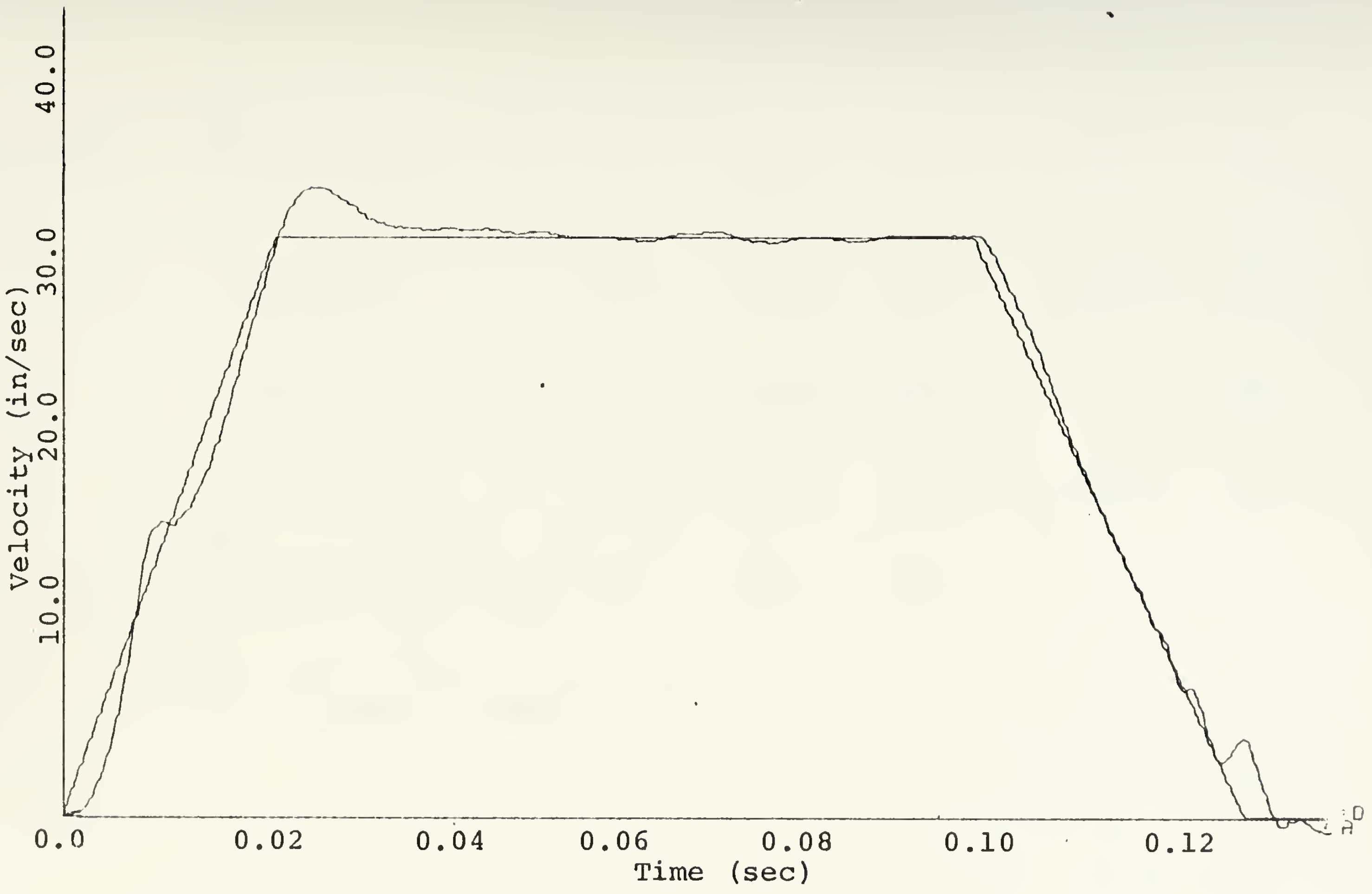


FIGURE 28 CADS Optimized System Response

Compensator	Cost Function	P_2	Z_2	P_3	Z_3	δ	W_n
Original		20.	50.	500.	50.	0.5	1300
C_I	$\int Err^2 dt$	21.	43.45	592	47.47	0.5	1300
C_N	$\int Err^2 dt$	21.	43.45	592	47.47	0.2	1500.
C_N	$\int \lvert Err \rvert * t dt$	21.	43.45	592	47.47	0.2	1430.
C_I, C_N	$\int_0^{0.045} \lvert Err \rvert dt + 4.14 \int_{0.045}^{0.13} \lvert Err \rvert dt$	27.1	52.5	521.	48.	0.32	1268.

TABLE II Summary of Servo Drive Optimization

This problem has presented an excellent example of the necessity to carefully select the cost function which will measure the system's performance. Defining a performance index which will weight the more objectionable characteristics of a system's response so that they are eliminated or reduced becomes difficult as the system's complexity increases. The performance index, $J = \int (Err)^2 \, dt$, was not adequate in its treatment of the noise and switching transients for the above problem. The other performance indexes used were also marginal in their effects upon parameter optimization. Although the cost function was reduced to a mathematically correct minimum, the system's performance was not the best that could be achieved. The system's performance was only optimum due to the definition of the cost function. The system was simulated using the compensator values determined from the last optimization run with the exception of δ which was increased to $\delta = 0.5$. This simulation was made based upon engineering judgment of the effects of varying δ. Figure 29 shows that this change in the damping factor produced a system response which was nearer the desired response than any of the previous runs.

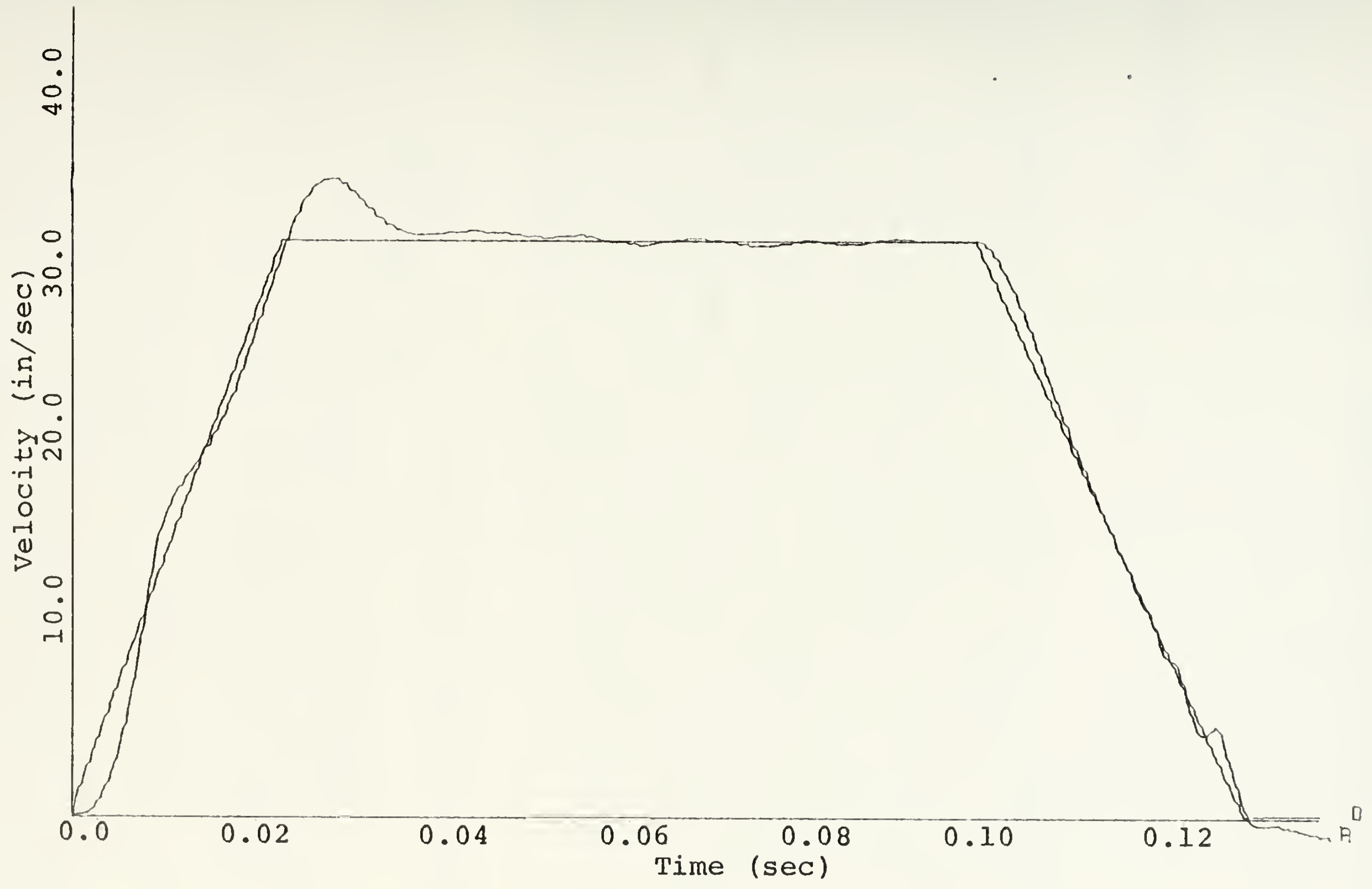


FIGURE 29 Servo System Response, CADS Values with $\delta = 0.5$

IV. DISCUSSION, CONCLUSIONS AND RECOMMENDATIONS

A. DISCUSSION

The objective of this thesis was to investigate the feasibility of developing a computer program which would optimize a variety of control systems' responses in the time domain. The program developed during the investigation proves that time domain optimization of control system responses is not only feasible but desirable due to the readily interpretable results of the optimization process. CADS requires approximately 200K bytes of computer core when the high speed printer plot is used for graphical display of the output and 230K when outputting calcomp graphs. These core requirements are a maximum and could be reduced by more careful, professional programming.

The computer time required for CADS to arrive at a solution is dependent upon

(1) the order of the system being simulated,

(2) the area of the search zone determined by the upper and lower bounds on the variable parameters and

(3) the nearness of the starting guess to the optimum parameter values.

Every trial value of a parameter selected by BOXPLX requires a complete simulation of the system in order to evaluate the system's response and compare it with the desired response. The program run time is therefore, T_{RUN} = the number of trials X system simulation time. A high order system may require twenty seconds of CPU time for simulation. If 300 trials are required to determine the optimum parameter values, the total CPU time would be 100 minutes.

Example Problem III	Lower Bound XL	Starting Guess XS	Upper Bound XU	Number of Trials	CPU Time Required
b (pseudo)	-300	-34.3	-3	55	7min03sec
c	.001	.01	0.1	225	24min53sec
	.01	.1	1.0	225	24min53sec
d	400	450	500	2,000	68min
	400	450	500		
	90	95	100		
	90	95	100		
	$3.4x10^6$	$3.5x10^6$	$3.5x10^6$		
e	10	20	30	230	4hr
	40	50	60		
	400	500	600		
	.3	.5	.55		
	1150	1300	1400		

Table III

Table III presents a summary of the search zones and times required to obtain a solution for some of the problems considered in this thesis. The amount of CPU time required for a solution, especially for problem III-E, may seem excessive. However, there are several considerations which should be made prior to arriving at this conclusion.

(1) The equations of the system do not have to be written, programmed nor debugged if the system's

component transfer functions are known.

(2) Time domain requirements do not have to be translated into frequency domain specifications for system simulation and design.

(3) A systematic search is carried out to obtain the optimum parameter settings. This assures that with valid bounds on the variables an acceptable solution will be obtained with the first optimization attempt.

The time required to perform the above steps in the design process by conventional means may result in many more hours of CPU time than if the program CADS were used from the beginning of the design process.

Several means of reducing the computer time required for optimization were previously outlined in section II. The most significant reduction is obtained by keeping the number of integrations required for system simulation to a minimum. Block diagram reduction of the system should be accomplished whenever possible. The example of the Ward Leonard drive system shown on page 26 can be reduced to the simple system shown in Figure 30 by block diagram reduction.

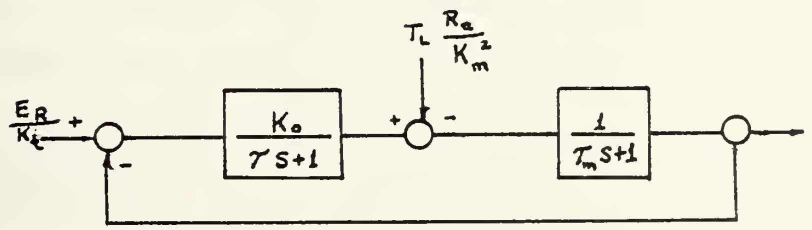


FIGURE 30 Block Reduced Ward-Leonard Drive

Section III.E. showed the effect of using different cost functions to measure the quality of the optimized response. Although the integral error squared criteria is often used to judge a system's performance, the user should carefully consider how to best define a cost function which will properly penalize deviations of the system response from the desired response. A small error (err < 1) squared becomes even smaller. If all the errors are small, the IES is a valid cost function but if the system response involves large and small errors a weighted cost function will have to be used. One method of arriving at a properly weighted cost function is to simulate the system using first estimates of the variable parameters and recording the sum of the errors over the different portions of the response. A ratio of the errors can then be used to arrive at proper weighting factors for each time section of the response.

BOXFIX will continue the optimization process, working in the seventh significant digit until it can no longer reduce the cost function. Often an acceptable solution for the system parameters has been found long before the cost function has been reduced to its minimum. A judicious use of CADS may be made by evaluating the system response after ten to fifteen minutes of run time to see if an acceptable solution has been found.

B. CONCLUSIONS

Time domain optimization using the CADS program is a simple, straight forward process which does not require an in-depth analysis of the system being optimized. Economic use of CPU times does dictate that intelligent starting values and bounds be placed on the variable parameters which are to be optimized.

The program is a readily usable tool for simulation without optimization. It is competitive with, if not superior to, other common simulation routines when simulating typical control systems. This feature alone is expected to bring the program into common usage.

C. RECOMMENDATIONS FOR FUTURE WORK

(1) All integral calculations performed during this investigation were done in double precision for increased accuracy of the system response. The possibility of using single precision calculations should be investigated to decrease core requirements.

(2) The ability to begin the optimization process at some time greater than zero should be provided. This will necessitate modification of the data input cards and block equations so that initial conditions can be entered.

(3) The program presently requires that external forcing functions (DRVINs) and the variables which are to be optimized be specified by Fortran IV statements placed within the body of the program. A method of reading these specifications from data card input should be developed so that the user will not have to "shuffle" cards in the program deck.

(4) The graphical output of CADS was all that was necessary for the investigations conducted in this thesis but a provision for numerical output of selected responses should be provided for detailed analysis.

(5) The feasibility of reducing the number of significant digits BCXPLX uses should be studied as a means of reducing optimization time. Also some criteria might be developed to reject a system's response before TF is reached if it is determined to be unacceptable.

(6) A method of automatically relaxing the boundaries on the variables being optimized when they go to their limits should be developed.

(7) The standard cost function provided and all user developed cost functions should be normalized. This would permit a more direct and easier comparison of a system's "goodness" when several different integration step sizes or run times have been used in the optimization process.

APPENDIX A

Block Data Card Format

```
|1        |11     |20     |40     |60

BLKCCD=EEVV       G       P       Z

CC = POSITION NUMBER OF THE BLOCK
D  = TYPE OF BLOCK (NUMBER)
EE = INPUT NODE NUMBER
VV = OUTPUT NODE NUMBER
G  = VALUE OF GAIN
P  = VALUE OF THE POLE*
Z  = VALUE OF THE ZERO*
```

* For block type 4, δ is read into the P location and Wn is read into the Z position.

* For block type 5, θ_U is read into the P position and θ_L is read into the Z position.

* For block type 6, θ_C is read into the P position and θ_δ is read into the Z position.

EXAMPLES

```
                     G

     BLKC11=0102     10.

                     G      P

     BLKC22=0304     1.     5.

                     G      P      Z

     BLKC33=0405     1.     10.    5.
```

	G		δ	ω_N
BLK104=1006	200.		.2	14.14
	G	θ_U	$-\theta_L$	
BLK115=1207	10.	20.	-5.	
	G	θ_C	θ_B	
BLK126=0708	1.	3.	10.	

*See Table I.

APPENDIX B

Plant Flow Chart

```
            FUNCTION PLANT (D)

            DIMENSION G(25), P(25), Z(25), FLAG(25,25),  OM
            G(25), THADOT(25), OMGDOT(25), DRVIN(25),
            NF(25), IR(25), IV(25), X2(25), X2DOT(25),
            DRIVE(25), THA(25), IC(25), ID(25), IE(25),
            THACUT(3001), XDATA(3001), D(25)

            REAL *8THACUT

            REAL *8XDATA

            REAL *8THA,THADOT,T,DT,DRIVE,OMG,OMGDOT,DRVIN,TF

            REAL *8H1,H2

            REAL *8X2,X2DOT

            COMMON T,DT,TF,THACUT,XDATA,M3,ICONT,NEQ,ISKIP,ITF
        FOR OPT RUNS, INHIBIT READ STATEMENTS AFTER READING
                          |
                     . *     * .
                . *      IF        * .
              *        ISKIP-1         *
          |   * .                    . *   |
          |       * .            . *       |
        - |            * . *               + |
                          0 |
       -------        -------         -------
       | 1   |        | 5   |         | 5   |
       -------        -------         -------
                          |
            ---------------------------------------
  1         | ISKIP=2                             |
            ---------------------------------------
                          |
         ***READ (5,24) N,ISET
        INITIALIZE COUNTERS
                          |
            ---------------------------------------
            | ICUT  =0                            |
            | IVCUT=0                             |
            | H1    =DT                           |
            | H2    =0.5D0*H1                     |
            | N11   =0                            |
            | N55   =0                            |
            | N66   =0                            |
            | ICK   =2*N                          |
            ---------------------------------------
                          |
      READ INPUT DATA
                       |
               +-------------+
              + DO            +
++++++++++   +    3            +
+             + I=1,N         +
+              +-------------+
+                      |
+       ***READ (5,25) IC(I),ID(I),IE(I),IV(I),G(I),P(I),Z(I)
+                      |
+      SET THRUUT = OUTPUT OF LAST BLOCK
+
+                      |
+                      |
+               . *       * .
+           . *     IF        * .
+         *(IV(I)-IVOUT).LE.0  *
+         * .                  . *----------------------------
+           * .              . *        T      | 2       |
+               *  .  *                        ---------
+                  |
+                  F
+                  |
+                  |
+       ------------------------------------------------------
+       | IVOUT=IV(I)                                       |
+       | IOUT =IC(I)                                       |
+       ------------------------------------------------------
+                  |
+                  |
+                  |
2 +             . *       * .
+           . *     IF        * .
+         *    IC(I).EQ.1      *
+         * .                  . *--------------------------------------
+           * .              . *        T      | N11=N11+1            |
+               *  .  *                        -----------------------
+                  |                                     |
+                  F                                     |
+                  |--------------------------------------
+                  |
+                  |
+               . *       * .
+           . *     IF        * .
+         *    IC(I).EQ.5      *
+         * .                  . *--------------------------------------
+           * .              . *        T      | N55=N55+1            |
+               *  .  *                        -----------------------
+                  |                                     |
+                  F                                     |
+                  |--------------------------------------
+                  |
+                  |
+               . *       * .
+           . *     IF        * .
+         *    IC(I).EQ.6      *
+         * .                  . *--------------------------------------
+           * .              . *        T      | N66=N66+1            |
+               *  .  *                        -----------------------
+                  |                                     |
+                  F                                     |
+                  |--------------------------------------
+                  |
3 ++++++***WRITE (6,26) IC(I),ID(I),IE(I),IV(I),G(I),P(I),Z(I)
```

```
     SET THACUT = SPECIFIED  THETA, IF ANY
                          |
                          |
                          |
                   . *    * .
               . *    IF      * .
             *     ISET.NE.0      *
             *  .              .  *----------------------------------
                 * .        . *          T       | ICUT=ISET           |
                    *  .  *                      ----------------------
                       |                              |
                       F                              |
                       |-------------------------------
                       |
      ***WRITE (6,27) ICUT
                       |
        ------------------------------------------------
        | N11   =4*N11                                 |
        | N55   =4*N55                                 |
        | N66   =4*N66                                 |
        | NEQ   =N-1                                   |
        ------------------------------------------------
                       |
     SCAN FOR INPUTS

     CONNECT SYSTEM BY SETTING FLAG=1. FOR CONNECTED BLKS
                       |
               +-------------+
               + DO          +
 ++++++++++    +    4         +
 +             + J=1,N       +
 +             +-------------+
 +                     |
 +                     |
 +             +-------------+
 +             + DO          +
 ++++++++++    +    4         +
 +             + K=1,N       +
 +             +-------------+
 +                     |
 +                     |
 +      ------------------------------------------------
 +      | FLAG(K,J) =0.0                               |
 +      ------------------------------------------------
 +                     |
 +                     |
 +                     |
4 +                . *    * .
 +             . *    IF      * .
 ++++++++*+ IV(K).EQ.IE(J)      *
 +           *  .              .  *----------------------------------
 +               * .        . *          T       | FLAG(K,J)=1.0       |
 +                  *  .  *                      ----------------------
 +                     |                              |
 +                     F                              |
 +                     |-------------------------------
 +                     |

     TAKE BLK TYPE AND SOLVE EQNS ONE BY ONE
```

```
      CLEARING CLT REGISTERS AND INITIALIZING COUNTERS
                              |
                   +-------------------+
 5                 + DO                  +
  +++++++++++               6              +
  +                + ICLR=1,N            +
  +                 +-------------------+
  +                           |
  +                           |
  +              ----------------------------------------
  +             | THA(ICLR) =0.D0                        |
  +             | THADOT(ICLR) =0.D0                     |
  +             | CMG(ICLR) =0.D0                        |
  +             | CMGDOT(ICLR) =0.D0                     |
  +             | DRVIN(ICLR) =0.D0                      |
  +             | X2(ICLR) =0.D0                         |
  +              ----------------------------------------
  +                           |
  +              ----------------------------------------
 6 +++++++++++ | X2DOT(ICLR) =0.D0                      |
                | T     =0.0D0                           |
                 ----------------------------------------
                              |
      NR'S CONTROL ENTRY POINT IN INTEGRATION SUBROUTINES
                              |
                 ----------------------------------------
                | NR2   =1                               |
                | NR3   =1                               |
                | NR4   =1                               |
                 ----------------------------------------
                              |
      M3 CONTROLS WHICH BLOCK EQUATION IS BEING SOLVED
                              |
                 ----------------------------------------
                | M3    =0                               |
                 ----------------------------------------
                              |
      ICONT IS USED TO CONTROL PROGRAM FLOW
                              |
                 ----------------------------------------
                | ICONT=0                                |
                 ----------------------------------------
                              |
      IWAIT IS USED TO CONTROL TIME.  TIME IS STEPPED EVERY

      FIRST AND THIRD PASS THRU INTEGRATION ROUTINES
                              |
                 ----------------------------------------
                | IWAIT=0                                |
                | IT    =0                               |
                 ----------------------------------------
                              |
      ILAST'S CONTROL PROGRAM DATA AND PROGRAM EXIT
                              |
                 ----------------------------------------
                | ILAST=0                                |
                | I5LAST =0                              |
                | I6LAST =0                              |
                 ----------------------------------------
                              |
        SPECIFY ANY PARAMETERS TO BE OPTIMIZED HERE
                          |
        EG.  P(1)=C(2),     Z(1)=C(1),

        SET DRIVES FOR INPUTS

                           |
                 +-------------+
  7             + DO            +
   ++++++++++  +     9           +
   +            + MDRV=1,N      +
   +             +-------------+
   +                       |
   +                       |
   +           ---------------------------------------
   +           | DRVIN(1) =1.                        |
   +           | DRIVE(MDRV) =0.D0                   |
   +           ---------------------------------------
   +                       |
   +                       |
   +             +-------------+
   +            + DO            +
   ++++++++++  +     8           +
   +            + M=1,N         +
   +             +-------------+
   +                       |
   +                       |
   +           ---------------------------------------
  8 ++++++++++ | DRIVE(MDRV) =DRIVE(MDRV)+THA(M)     |
   +           |        *FLAG(M,MDRV)                |
   +           ---------------------------------------
   +                       |
   +           ---------------------------------------
  9 ++++++++++ | DRIVE(MDRV) =DRVIN(MDRV)+DRIVE(MDRV) |
               ---------------------------------------
                           |
               ---------------------------------------
 10            | M3    =M3+1                         |
               ---------------------------------------
                           |
        PICK TYPE EQN TO SOLVE

                           |
                     . *     * .
                . *      IF       * .
              *     IWAIT.EQ.0        *
              * .                   . *----------------------------
                  * .           . *          T     | T=T+H2       |
                       *   .   *                   ----------------
                           |                             |
                           F                             |
                           |------------------------------
                           |
                           |
                     . *     * .
                . *      IF       * .
              *     ID(M3).EQ.1       *
              * .                   . *--------------------
                  * .           . *          T     | 11   |
                       *   .   *                   --------
                           |
                           F
                           |

                              73
```

```
                        |
                        |
                 .  *       *  .
            .  *       IF        *  .
         *          ID(M3).EQ.2          *
         *  .                          .  *-------------------
            *  .                    .  *          T      | 12   |
                 *     .     *                           -------
                       |
                       F
                       |
                       |
                       |
                 .  *       *  .
            .  *       IF        *  .
         *          ID(M3).EQ.3          *
         *  .                          .  *-------------------
            *  .                    .  *          T      | 13   |
                 *     .     *                           -------
                       |
                       F
                       |
                       |
                       |
                 .  *       *  .
            .  *       IF        *  .
         *          ID(M3).EQ.4          *
         *  .                          .  *-------------------
            *  .                    .  *          T      | 14   |
                 *     .     *                           -------
                       |
                       F
                       |
                       |
                       |
                 .  *       *  .
            .  *       IF        *  .
         *          ID(M3).EQ.5          *
         *  .                          .  *-------------------
            *  .                    .  *          T      | 15   |
                 *     .     *                           -------
                       |
                       F
                       |
                       |
                       |
                 .  *       *  .
            .  *       IF        *  .
         *          ID(M3).EQ.6          *
         *  .                          .  *-------------------
            *  .                    .  *          T      | 16   |
                 *     .     *                           -------
                       |
                       F
                       |
                       |
***WRITE (6,28)
   STOP                |
        START SOLUTION

        TYPE ONE EQUATIONS

        SOLVES  THAOUT = G*THAIN

                          |
           ------------------------------------------------
11         | THA(M3) =G(M3)*DRIVE(M3)                     |
           | IWAIT=IWAIT+1                                |
           | ILAST=ILAST+1                                |
           ------------------------------------------------
                          |
                          |
                          |
                  . *         * .
                . *       IF      * .
          *(ILAST.EQ.N11).AND.(ICONT.EQ.NEQ)
          * .                          . *------------------
              * .                 . *        T     | 21   |
                   *      .     *                  -------
                          |
                          F
                          |
                          |
                      -------
                      | 18  |
                      -------


        TYPE TWO EQNS

        SOLVES THAOUT/THAIN = G/(S + P)

                          |
           ------------------------------------------------
12         | THADOT(M3) =-P(M3)*THA(M3)+G(M3)             |
           |       *DRIVE(M3)                             |
           | S     =RKLDE2(THA,THADOT,NR2)                |
           | IWAIT=IWAIT+1                                |
           ------------------------------------------------
                          |
                          |
                          |
                  . *         * .
              . *       IF        * .
          *             S-1.0          *
       |  * .                        . *  |
       |      * .                 . *     |
       |           *      .     *         |
     - |               0  |             + |
    -------          -------          -------
    | 17  |          | 18  |          | 21  |
    -------          -------          -------
                          |
        TYPE THREE EQNS

        SOLVES  THAOUT/THAIN = G*(S + Z)/(S + P)

                          |
           ------------------------------------------------
13         | OMGDOT(M3) =-P(M3)*OMG(M3)+G(M3)             |
           |       *DRIVE(M3)                             |
           ------------------------------------------------
                          |
                    |
         S=RKLDE3(OMG,OMGDOT,NR3)
                    |
         -------------------------------------------
         | THA(M3) =(Z(M3)-P(M3))*OMG(M3)          |
         |       +G(M3)*DRIVE(M3)                  |
         | IWAIT=IWAIT+1                           |
         -------------------------------------------
                    |
                    |
                    |
               . *     * .
           . *     IF      * .
         *        S-1.0    .   *
     |   *  .              .  *    |
     |        * .        . *       |
     |            *  .  *          |
   - |             0 |          +  |
  -------       -------       -------
  | 17  |       | 18  |       | 21  |
  -------       -------       -------
                    |

    TYPE FOUR EQNS.

    SOLVES  THAOUT/THAIN = G/(S**2 + 2*DELTA*WN*S + WN**2)
                    |
         -------------------------------------------
14       | V       =COPLX(P,Z,G,DRIVE,X2,OMG,NR4)  |
         | THA(M3) =OMG(M3)                        |
         | IWAIT=IWAIT+1                           |
         -------------------------------------------
                    |
                    |
                    |
               . *     * .
           . *     IF      * .
         *        V-1.         *
     |   *  .              .  *    |
     |        * .        . *       |
     |            *  .  *          |
   - |             0 |          +  |
  -------       -------       -------
  | 17  |       | 18  |       | 21  |
  -------       -------       -------
                    |

    TYPE FIVE EQNS

    SOLVES  THAOUT = G*THAIN  FOR /G*THAIN/ < SAT LIMITS
                    |
         -------------------------------------------
15       | THA(M3) =G(M3)*DRIVE(M3)                |
         -------------------------------------------
                    |
                    |
                    |
               . *     * .
           . *     IF      * .
         * THA(M3).GT.P(M3)    *
         *  .              .  *------------------------------
              * .        . *         T    | THA(M3)=P(M3)   |
                  *  .  *                 -------------------
                     |                           |
                     F                           |
                     |----------------------------
                     |
```

76

```
                     |
                     |
             . *     *  .
          . *     IF      * .
        * THA(M3).LT.Z(M3)   *
        * .                . *-------------------------------
            * .         . *         T      | THA(M3)=Z(M3)     |
               *  .  *                     --------------------
                  |                              |
                  F                              |
                  |-------------------------------
                  |
                  |
     ----------------------------------------------
     | IWAIT=IWAIT+1                               |
     | I5LAST =I5LAST+1                            |
     ----------------------------------------------
                  |
                  |
             . *     *  .
          . *     IF      * .
        *(I5LAST.EQ.N55).AND.(ICONT.EQ.NEG)
        * .                . *---------------------
            * .         . *         T      | 21  |
               *  .  *                     -------
                  |
                  F
                  |
                  |
               -------
               | 18  |
               -------


  .   TYPE SIX EQNS

      SOLVES  THAOUT = G*THAIN  FOR CONT REF > D Z BOUND
                  |
     ----------------------------------------------
16   | THA(M3) =G(M3)*DRIVE(M3)                    |
     | IP    =P(M3)                                |
     ----------------------------------------------
                  |
                  |
             . *     *  .
          . *     IF      * .
        *(DABS(THA(IP))).LT.Z(M3)
        * .                . *-----------------------------
            * .         . *         T      | THA(M3)=0.      |
               *  .  *                     ------------------
                  |                              |
                  F                              |
                  |-------------------------------
                  |
                  |
     ----------------------------------------------
     | IWAIT=IWAIT+1                               |
     | I6LAST =I6LAST+1                            |
     ----------------------------------------------
                  |
                    |
               . *     * .
            . *    IF      * .
         *(IELAST.EQ.NEQ).AND.(ICONT.EQ.NEQ)
         * .                     . *-----------------
            * .               . *       T     | 21   |
               *  .  *                        -------
                  |
                  F
                  |
                  |
               -------
               | 18  |
               -------


17       ***WRITE (6,29)
            STOP
                  |
         ------------------------------------------
18       | ICONT=ICONT+1                           |
         ------------------------------------------
                  |
                  |
               . *     * .
            . *    IF      * .
     |   *       ICONT-N        *   |
     |   * .                  . *   |
     |      * .            . *      |
   - |         *  .  *              + |
               0  |
   -------     -------     -------
   | 10  |     | 19  |     | 20  |
   -------     -------     -------
                  |
       ONE PASS HAS BEEN MADE FOR EACH EQN.  INCREMENT
                  |
         ------------------------------------------
19       | NR2   =NR2+1                            |
         | NR3   =NR3+1                            |
         | NR4   =NR4+1                            |
         | M3    =0                                |
         | ICONT=0                                 |
         ------------------------------------------
                  |
                  |
               . *     * .
            . *    IF      * .
         *   IWAIT.EQ.ICK       *
         * .                  . *-----------------------
            * .            . *        T     | IWAIT=0              |
               *  .  *                      -----------------------
                  |                                |
                  F                                |
                  |---------------------------------
               -------
               | 7   |
               -------


20       ***WRITE (6,30)
            STOP
```

```
          DATA COLLECTION POINT.  DATA IS RECORDED AT END OF

          COMPLETE TIME STEP(DT)
                                 |
             -----------------------------------------------
21           | IT    =IT+1                                 |
             | THACUT(IT) =THA(IOUT)                       |
             -----------------------------------------------
                                 |
          TEST FOR END OF RUN
                                 |
                                 |
                         .  *       *  .
                    .  *      IF        *  .
                 *           T-TF             *
              |  *  .                      .  *  |
              |        *  .          .  *        |
              |              *  .  *             |
           -  |              0  |             +  |
          --------        --------        --------
          | 22   |        | 22   |        | 23   |
          --------        --------        --------
                                 |
          RESET COUNTERS FOR NEXT PASS THRU EQUATIONS
                                 |
             -----------------------------------------------
22           | NR2   =1                                    |
             | NR3   =1                                    |
             | NR4   =1                                    |
             | N3    =0                                    |
             | ICNT=0                                      |
             | IWAIT=0                                     |
             | ILAST=0                                     |
             | I5LAST =0                                   |
             | I6LAST =0                                   |
             -----------------------------------------------
                                 |
                                 |
                          --------
                          | 7    |
                          --------
                                 |
             -----------------------------------------------
23           | PLANT=1.                                    |
             -----------------------------------------------
                                 |

             RETURN
24           FORMAT (I2,2X,I2)
25           FORMAT (3X,I2,I1,1X,2I2,8X,3E20.7)
26           FORMAT (/,5H BLK ,I2,I1,2H =,2I2,6X,3E20.7)
27           FORMAT (//,2X,'THETA OUT IS THA(',I2,')')
28           FORMAT (40H *** EQN SWITCH CONTROL DID NOT WORK)
29           FORMAT (20H INTEGRATION TROUBLE)
30           FORMAT (58H ERROR IN INTERGATION. ATTEMPTED TO
             INTEG. MORE THAN N-EQN)
END
```

```
C
C     DECK MODIFIED FOR PROBLEM III-E.
C
C     MODIFICATIONS TO MAIN
C
C     CALCULATE STANDARD SECOND ORDER SYSTEM RESPONSE FOR BOXPLS DATA
C          XDD = -2*&*WN*XD - WN**2*X + DRV
C
      IDATA = 0
      X(1) = 0.D0
C
C     DO NOT COMPUTE DATA CURVE IF LEAP = 1
C
      IF (LEAP.EQ.1) GO TO 5
      X(2) = 0.D0
      NT = 0
  181 T=T+DT
      IF(T.GE.0.13) GO TO 1813
      IF(T.GE.0.1) GO TO 1812
      IF(T.GE.0.0236) GO TO 1811
      X(1)=1347.46*T
      GO TO 1814
 1811 X(1)=31.8
      GO TO 1814
 1812 X(1)=31.8-1060.*(T-0.1)
      GO TO 1814
 1813 X(1)=0.
 1814 IDATA=IDATA+1
      XDATA(IDATA)=X(1)
      IF(T-TF)181,183,183
C     NOW HAVE STANDARD VALUES    CONTINUE WITH PROB
  183 CONTINUE
C
C     MODIFICATIONS TO PLANT
C
C
    7 DO 9 MDRV=1,N
C
C     DRVIN(I)'S GO HERE IF FUNCTIONS OF TIME
C
 8888 THA12=THA(12)
      INDRV=INT(THA12)
      DRVIN(2)=FLOAT(INDRV)
      IF(IWAIT.EQ.0) T=T+H2
      IF(T.GE.0.13) GO TO 1122
      IF(T.GE.0.1) GO TO 1123
      IF(T.GE.0.0236) GO TO 1124
      DRVIN(1)=1347.46*T
      GO TO 1125
 1124 DRVIN(1)=31.8
      GO TO 1125
 1123 DRVIN(1)=31.8-1060.*(T-0.1)
      GO TO 1125
 1122 DRVIN(1)=0.0
 1125 CONTINUE
      IF(T.LT.0.0236) GO TO 1126
      IF(T.GT.0.1) GO TO 1127
      GO TO 1128
 1126 DRVIN(5)=2.809
      GO TO 1129
 1127 IF(T.GT.0.13) GO TO 1128
      DRVIN(5)=-2.809
      GO TO 1129
 1128 DRVIN(5)=0.
 1129 CONTINUE
      IF(T.LT.0.0236) GO TO 1130
      IF(T.GT.0.1) GO TO 1131
      DRVIN(4)=0.9839
      GO TO 1132
 1130 DRVIN(4)=41.6922*T
      GO TO 1132
 1131 IF(T.GT.0.13) GO TO 1133
      DRVIN(4)=0.9839+32.7978*(0.1-T)
      GO TO 1132
 1133 DRVIN(4)=0.
 1132 CONTINUE
C
      DRIVE(MDRV) = 0.D0
C
      DO 8 M=1,N
    8 DRIVE(MDRV) = DRIVE(MDRV)+THA(M)*FLAG(M,MDRV)
C
    9 DRIVE(MDRV) = DRVIN(MDRV)+DRIVE(MDRV)
C
```

```
//   EXEC  FORTCLG,REGION.GO=200K
//FORT.SYSIN DD *
C    ********************************************************************
C    *                                                                  *
C    *                                                                  *
C    *              COMPUTER AUTOMATED DESIGN OF SYSTEMS                *
C    *                             (CACS)                               *
C    *                                                                  *
C    *                              BY                                  *
C    *                       LARRY PAUL VINES                           *
C    *                        LIEUTENENT USN                            *
C    *                                                                  *
C    *                           1 APRIL 76                             *
C    *                                                                  *
C    *                                                                  *
C    * THIS PROGRAM WILL SIMULATE/OPTIMIZE CONTROL SYSTEMS AND CIRCUITS.*
C    * THE INPUT DATA IS IN TRANSFER FUNCTION BLOCK FORM.  TO SIMULATE  *
C    * THE SYSTEM WHICH IS TO BE OPTIMIZED, TRANSFER FUNCTIONS WHICH    *
C    * ARE COMMONLY ENCOUNTERED WERE REDUCED TO FIRST ORDER DIFFERENTIAL*
C    * EQUATIONS.  THESE EQUATIONS WERE THEN PROGRAMMED SO THAT THE     *
C    * TRANSFER FUNCTION BLOCKS COULD BE CONNECTED IN AN ARBITRARY      *
C    * FASHION FROM THE DATA CARD INPUT.  SEVERAL COMMON NONLINEAR      *
C    * TRANSFER BLOCKS WERE ALSO PROVIDED.  THESE BLOCKS SHOULD BE      *
C    * ADEQUATE TO SIMULATE MOST CONTROL SYSTEMS.  THE PROGRAM WILL     *
C    * ALLOW FOR A WIDE VARITY OF CONTROL SYSTEMS TO BE EASILY          *
C    * SIMULATED WITH THE KNOWN SYSTEM PARAMETERS AND THEN ALLOW        *
C    * ALL UNKNOWN OR ADJUSTABLE PARAMETERS TO BE SET BY THE            *
C    * MINIMIZATION ROUTINE(BOXPLX) TO ACHIEVE THE DESIRED RESPONSE.    *
C    * A GRAPHICAL OUTPUT OF THE DESIRED RESPONSE AND ACTUAL SYSTEM     *
C    * RESPONSE IS THEN PROVIDED.                                       *
C    *                                                                  *
C    *                                                                  *
C    * DESCRIPTION OF PARAMETERS                                        *
C    *                                                                  *
C    *                                                                  *
C    * NRUNS = NUMBER OF RUNS TO BE MADE.  NRUNS = 1 WHEN OPTIMIZING.   *
C    *                                                                  *
C    * NV = NUMBER OF VARIABLES BOXPLX WILL SET.                        *
C    *                                                                  *
C    * NAV = NUMBER OF AUX VARIABLES DEFINED.                           *
C    *                                                                  *
C    * LEAP = 0 COMPUTES STANDARD SECOND ORDER STEP RESPONSE FOR        *
C    *          DATA CURVE.                                             *
C    *      = 1 SKIPS SECOND  ORDER DATA EQUATION                       *
C    *                                                                  *
C    * NPR = FREQUENCY OF OUTPUT FROM BOXPLX FOR DIAGNOSTIC PURPOSES.   *
C    *       (NPR = 25, 50, 100 IS RECOMMENDED)                         *
C    *                                                                  *
```

```
C     * NTA = NUMBER OF TRAILS ALLOWED BY BOXPLX FOR A SOLUTION.             *        470
C     *                                                                      *        480
C     * IPLOT = OPTIONAL PRINTER PLOT OF SOLUTION.                           *        490
C     *         = 1 PRODUCES PLOT.   = 0  NO PLOT.                           *        500
C     *                                                                      *        510
C     * IDRAW = OPTIONAL CALCOMP GRAPH OF SOLUTION.                          *        520
C     *         = 1  PRODUCES PLOT.   = 0  NO PLOT                           *        530
C     * *** ONLY ONE PLOT OPTION MAY BE USED AT A TIME.  IDRAW REQUIRES      *        540
C     *     A //EXEC CLGP  CONTROL CARD AND 230K REGION.                     *        550
C     *                                                                      *        560
C     * T = TIME INITIAL CONDITION.  MUST BE ZERO.                           *        570
C     * DT = STEP SIZE FOR INTEGRATION.  TF/1000. IS RECOMMENDED.            *        580
C     * TF = FINAL PROBLEM TIME.                                             *        590
C     *                                                                      *        600
C     * DELTA = DAMPING FACTOR                                               *        610
C     * WN = NATURAL FREQUENCY    FOR STANDARD 2ND ORDER DATA CURVE          *        620
C     * BETA = GAIN FACTOR                                                   *        630
C     *                                                                      *        640
C     * XS(I) = STARTING GUESS                                               *        650
C     * XU(I) = UPPER BOUND      FOR  VARIABLE PARAMETERS TO BE SET BY       *        660
C     * XL(I) = LOWER BOUND      THE MINIMIZATION ROUTINE                    *        670
C     *                                                                      *        680
C     * N = NUMBER OF TRANSFER FUNCTION BLOCKS CONNECTED.                    *        690
C     *                                                                      *        700
C     * ISET = OUTPUT VARIABLE TO BE OPTIMIZED/PLOTTED.                      *        710
C     *        DEFAULT VARIABLE PLOTTED IS HIGHEST NUMBERED NODE VAR.        *        720
C     *                                                                      *        730
C     *                                                                      *        740
C     *                                                                      *        750
C     * DESCRIPTION OF BLOCK TRANSFER FUNCTIONS                              *        760
C     *                                                                      *        770
C     *                                                                      *        780
C     * BLKCOD=EEVV     G      P/DELTA/UL/REF      Z/WN/LL/BOUND             *        790
C     *                                                                      *        800
C     * CC = NUMBER OF BLOCK IN SYSTEM CONFIGURATION (IE. BLK NUMBER 1)      *        810
C     *                                                                      *        820
C     * D = TYPE OF TRANSFER FUNCTION BLOCK.                                 *        830
C     *                                                                      *        840
C     * EE = INPUT NODE NUMBER                                               *        850
C     *                                                                      *        860
C     * VV = OUTPUT NODE NUMBER                                              *        870
C     *                                                                      *        880
C     * G = GAIN OF BLOCK                                                    *        890
C     *                                                                      *        900
C     * P = POLE OF BLOCK TRANSFER FUNCTION                                  *        910
C     *                                                                      *        920
C     * DELTA = DAMPING FACTOR FOR COMPLEX TRANSFER BLOCK (TYPE 4)           *        930
C     *                                                                      *        940
```

```
C    * UL = UPPER LIMIT FOR LIMIT BLOCK (TYPE 5)                              *
C    *                                                                        *
C    * REF = REFERENCE FOR DEAD ZONE BLOCK (TYPE 6)                           *
C    *                                                                        *
C    * Z = ZERO FOR BLOCK TRANSFER FUNCTION                                   *
C    *                                                                        *
C    * WN = NATURAL FREQUENCY FOR COMPLEX BLOCK (TYPE 4)                      *
C    *                                                                        *
C    * LL = LOWER LIMIT FOR LIMIT BLOCK (TYPE 5)                              *
C    *                                                                        *
C    * BOUND = MAGNITUDE OF DEAD ZONE (TYPE 6)                                *
C    *                                                                        *
C    * C(I) = AUXILIARY VARIABLES (USED IN BOXPLX)                            *
C    *                                                                        *
C    * DRVIN(I) = EXTERNAL FORCING INPUTS                                     *
C    *                                                                        *
C    *                                                                        *
C    * DATA CARD FORMATS                                                      *
C    *                                                                        *
C    *                                                                        *
C    * CARD 1  NRUNS (I1)                                                     *
C    *                                                                        *
C    * CARD 2  NV,NAV,LEAP,NPR,NTA,IPLOT,IDRAW (7I5)                          *
C    *                                                                        *
C    * CARD 3  T,DT,TF (3F10.5)                                               *
C    *                                                                        *
C    * CARD 4  DELTA, WN, BETA (3F10.5)                                       *
C    *                                                                        *
C    * STARTING GUESS AND LIMIT CARDS (USED ONLY WHEN OPTIMIZING)             *
C    *                                                                        *
C    *                 XS(I)                                                  *
C    *                 XU(I)   N(E15.7,5X)   N MAX = 25                       *
C    *                 XL(I)                                                  *
C    *                                                                        *
C    * CARD 5  N,ISET  (I2,2X,I2)                                             *
C    *                                                                        *
C    * CARD 6  BLK(I2,I1=I2,I2, 8X 3E20.7)                                    *
C    *         AS MANY CARDS AS THERE ARE BLOCKS CONNECTED                    *
C    *                                                                        *
C    * ..... TWO TITLE CARDS ARE REQUIRED IF CALCOMP PLOTS                    *
C    *         ARE TO BE OUTPUT (COL 1-48)                                    *
C    *                                                                        *
C    *                                                                        *
C    **************************************************************************
C
C
      DIMENSION XS(25), XL(25), XU(25)
      DIMENSION X(2), XDOT(2)
      DIMENSION THAOUT(3001), XDATA(3001)
      REAL *8XDATA
      REAL *8THAOUT
      REAL *8X,XDOT,T,TF,DT
      REAL *8TF
      COMMON T,DT,TF,THAOUT,XDATA,M3,ICCNT,NEQ,ISKIP,ITF
C
C     THE MULTPLE RUN OPTION SHOULD NOT BE USED WHEN EMPLOYING
C     BOXPLX.  TIME BECOMES EXCESSIVE.  USE ONLY FOR PLANT - DATA RUNS
C
      READ (5,12) NRUNS
C
C     READ/WRITE CONTROL DATA
C
C
      DO 11 IRUN=1,NRUNS
      READ (5,13) NV,NAV,LEAP,NPR,NTA,IPLOT,IDRAW
      WRITE (6,14) NV,NAV,LEAP,NPR,NTA
      READ (5,15) T,DT,TF
      WRITE (6,16) T,DT,TF
      READ (5,15) DELTA,WN,BETA
      WRITE (6,17) DELTA,WN,BETA
C
C     READ/WRITE SEARCH BOUNDS IF OPTIMIZING
C
      IF (NV.EQ.0) GO TO 1
C
C     SET LIMITS
C
C
C     STARTING GUESS
      READ (5,18) (XS(I),I=1,NV)
C
C     UPPER BOUND
      READ (5,18) (XU(I),I=1,NV)
C
C     LOWER BOUND
      READ (5,18) (XL(I),I=1,NV)
      WRITE (6,19)
      WRITE (6,20) (XS(I),I=1,NV)
      WRITE (6,21)
      WRITE (6,20) (XU(I),I=1,NV)
      WRITE (6,22)
      WRITE (6,20) (XL(I),I=1,NV)
    1 CONTINUE
      IDP = TF/DT
      ITF = IDP
      IF (IDP.GT.4500) IDP=4500
C
```

```
C      PLCT EVERY FIFTH CATA PCINT
C
       NFFLT = 0.2*IDP
       R = 1./3.
       IF = 0
       ISKIP = 0
C
C
C      CALCULATE STANDARC SECONC ORCER SYSTEM RESPCNSE FCR BOXFLS CATA
C            XCC = -2*&*WN*XD - WN**2*X + DRV
C
       ICATA = 0
       X(1) = 0.C0
C
C      CC NCT COMPUTE DATA CURVE IF LEAF = 1
C
       IF (LEAF.EC.1) GO TO 5
       X(2) = 0.D0
       NT = 0
     2 XCCT(1) = X(2)
       XCCT(2) = -2.*DELTA*WN*X(2)-WN**2*X(1)+BETA
       S = RKLCEQ(2,X,XDOT,T,CT,NT)
       IF (S-1.) 3,2,4
     3 WRITE (6,23)
       STCP
C
     4 ICATA = IDATA+1
       XCATA(ICATA) = X(1)
C
C      TEST FOR END OF CCMPUTATION
C
       IF (T-TF) 2,5,5
C      NCW HAVE STANDARD VALUES    CONTINUE WITH PROB
     5 CCNTINUE
C
C      SET CATA CURVE = 0. IF NOT PLOTTING
C
       IF (LEAF.EC.0) GO TC 7
C
       CC 6 I=1,4500
       ICATA = IDATA+1
       T = T+DT
       XCATA(ICATA) = X(1)
       IF (T-TF) 6,7,7
     6 CCNTINUE
C
C
C      CALL PLANT TO SIMULATE THE SYSTEM
```

```
C
    7 IF (NV.EC.0) PL = PLANT(1.)
      IF (NV.EC.0) GO TO 8
C
C     IF CPTIMIZING, CALL BCXPLX AND WRITE OPTIMIZED VALLES
C
      CALL BOXPLX (NV,NAV,NPR,NTA,R,XS,IP,XU,XL,YMN,IER)
      WRITE (6,24) (XS(I),I=1,NV)
      WRITE (6,25) YMN,IER
C
C     PLCT SYSTEM RESPONSE
C
    8 IF (IPLCT.EC.0) GO TO 9
      WRITE (6,26)
      CALL FPLT (XDATA,THAOUT,DT,NPPLT,IDP)
      GC TO 11
    9 IF (ICRAW.EQ.0) GO TO 10
      CALL PIC (XDATA,THACUT,DT,NPPLT,ICP)
      GC TO 11
   10 WRITE (6,27)
   11 WRITE (6,28) IRUN
C
      STCP
C
   12 FCRMAT (I1)
   13 FCRMAT (7I5)
   14 FCRMAT (//,2X,'NV = ',I5,2X,'NAV = ',I5,2X,'LEAP = ',I1,2X,'NPR =
     1',I5,2X,'NTA = ',I5)
   15 FCRMAT (3F10.5)
   16 FCRMAT (//,2X,'T(0)=',F7.3,2X,'CT=',F10.5,2X,'TF=',F10.5)
   17 FCRMAT (/,2X,'DELTA=',F10.5,2X,'WN=',F10.5,2X,'BETA=',F10.5)
   18 FCRMAT (4(E15.7,5X))
   19 FCRMAT (/,2X,'STARTING GUESS')
   20 FCRMAT (//,2X,5(E15.7,5X))
   21 FCRMAT (/,2X,'UPPER BCUNDS')
   22 FCRMAT (/,2X,'LOWER BCUNDS')
   23 FORMAT (5X,'DID NCT DO STANCARD ECN PROPERLY  STOPEC INT')
   24 FCRMAT (///,'  THE XS(I) ARE  ',5(E15.7,5X))
   25 FCRMAT (//,2X,'THE FUNCTION VALLE IS   ',E13.7,2X,'IER = ',I5)
   26 FCRMAT ('1')
   27 FCRMAT (/,'* * NO PLOTS CALLED FCR * *')
   28 FCRMAT (//,2X,'*** RUN ',I1,' CCMPLETE ***')
      ENC
C
C     KE EVALLATES IMPLICIT CCNSTRAINTS CN BOXPLX VARIABLES
      FLNCTICN KE (C)
      CIMENSICN C(25)
C
```

```
C     NO IMPLICIT CONSTRAINTS
      KE = 0
      RETURN
      END
C
C     FE COMPUTES THE ERROR FUNCTION(PERFORMANCE INDEX) FOR BOXPLX
      FUNCTION FE (C)
      DIMENSION THAOUT(3001), XDATA(3001), C(25)
      REAL *8XDATA,THAOUT,DT,T,DIFF,PI
      REAL *8TF
      COMMON T,DT,TF,THAOUT,XDATA,M3,ICCNT,NEQ,ISKIP,ITF
      THAOUT(1) = 0.D0
      DIFF = 0.D0
      PI = 0.D0
      PL = PLANT(C)
C
      DO 1 I=1,ITF
      DIFF = XDATA(I)-THAOUT(I)
    1 PI = PI+DIFF**2
C
C
C     VALUE OF P I
      FE = PI
      RETURN
      END
C
C     FUNCTION PLANT SIMULATES THE SYSTEM
C
      FUNCTION PLANT (C)
      DIMENSION G(25), P(25), Z(25), FLAG(25,25), DRIVE(25), THA(25), OM
     1G(25), THADOT(25), OMGDOT(25), DRVIN(25), IC(25), ID(25), IE(25),
     2NF(25), IR(25), IV(25), X2(25), X2DOT(25), THAOUT(3001), XDATA(300
     31), C(25)
      REAL *8THAOUT
      REAL *8XDATA
      REAL *8THA,THADOT,T,DT,DRIVE,OMG,OMGDOT,DRVIN,TF
      REAL *8H1,H2
      REAL *8X2,X2DOT
      COMMON T,DT,TF,THAOUT,XDATA,M3,ICCNT,NEQ,ISKIP,ITF
C
C     FOR OPTIMIZATION RUNS INHIBIT READ STATEMENTS AFTER READING
C
      IF (ISKIP-1) 1,5,5
    1 ISKIP = 2
      READ (5,24) N,ISET
C
C     INITIALIZE COUNTERS
C
```

```
      IOUT = 0
      IVOUT = 0
      H1 = DT
      H2 = 0.5D0*H1
      N11 = 0
      N55 = 0
      N66 = 0
      ICK = 2*N
C
C     ***READ INPUT DATA***
C     COMPLEX BLOCK TYPE 44 READS DELTA IN FROM P FIELD AND
C     WN IS READ IN Z FIELD.
C     SATURATION BLOCK TYPE 55 READS PLUS SAT LEVEL IN FROM P FIELD AND
C     NEG SAT LEVEL FROM Z FIELD.
C     DEAD ZONE BLOCK TYPE 6  READS CONTROL REF FROM P FIELD AND
C     DEAD ZONE BOUND FROM Z FIELD
C
C
      DO 3 I=1,N
      READ (5,25) IC(I),ID(I),IE(I),IV(I),G(I),P(I),Z(I)
C
C     SET THACUT = OUTPUT OF LAST BLOCK
C
      IF ((IV(I)-IVOUT).LE.0) GO TO 2
      IVOUT = IV(I)
      IOUT = IC(I)
    2 IF (IC(I).EQ.1) N11=N11+1
      IF (IC(I).EQ.5) N55=N55+1
      IF (IC(I).EQ.6) N66=N66+1
C
C
    3 WRITE (6,26) IC(I),ID(I),IE(I),IV(I),G(I),P(I),Z(I)
C
C
C     SET THACUT = SPECIFIED  THETA, IF ANY
C
      IF (ISET.NE.0) IOUT=ISET
      WRITE (6,27) IOUT
      N11 = 4*N11
      N55 = 4*N55
      N66 = 4*N66
C
C     *** SHOULD NOW HAVE INPUT DATA AVAILABLE IN C,D,E,V ***
C
      NEC = N-1
C
C     *** SCAN FOR INPUTS ***
C
```

```
C     CONNECT UP SYSTEM BY SETTING FLAG=1. IF BLOCKS ARE CONNECTED
C
C
      DO 4 J=1,N
C
      DO 4 K=1,N
      FLAG(K,J) = 0.0
    4 IF (IV(K).EQ.IE(J)) FLAG(K,J)=1.0
C
C
C     ***NOW MUST TAKE TYPE OF BLK AND GO TO EQNS FOR SOLUTION***
C         BY SOLVING BLOCK BY BLOCK
C
C     CLEARING OUT REGISTERS AND INITIALIZING COUNTERS
C
C
    5 DO 6 ICLR=1,N
      THA(ICLR) = 0.D0
      THADOT(ICLR) = 0.D0
      CNG(ICLR) = 0.D0
      CNGDOT(ICLR) = 0.D0
      CFVIN(ICLR) = 0.D0
      X2(ICLR) = 0.D0
    6 X2DOT(ICLR) = 0.D0
C
      T = 0.0D0
C
C     NF'S CONTROL ENTRY POINT IN INTEGRATION SUBROUTINES(RKLDE'S)
C
      NF2 = 1
      NF3 = 1
      NF4 = 1
C
C     N3 CONTROLS WHICH BLOCK EQUATION IS BEING SOLVED
C
      N3 = 0
C
C     ICNT IS USED TO CONTROL PROGRAM FLOW
C
      ICNT = 0
C
C     IWAIT IS USED TO CONTROL TIME.  TIME IS STEPPED EVERY FIRST AND
C     THIRD PASS THRU INTEGRATION ROUTINES
C
      IWAIT = 0
      IT = 0
C
C     ILAST'S CONTROL PROGRAM DATA AND PROGRAM EXIT
```

```
C
      ILAST = 0
      I5LAST = 0
      I6LAST = 0
C
C
C     SPECIFY ANY PARAMETERS TO BE OPTIMIZED
C     EG.  P(1)=C(2),      Z(1)=C(1),
C
C     *** INSERT NV VARIABLE SPECIFICATIONS HERE ***
      G(3)=C(1)
C
C     ***SET DRIVES FOR INPUTS***
      DRVIN(1)=1.
C
    7 DO 9 MDRV=1,N
C
C     DRVIN(I)'S GO HERE IF FUNCTIONS OF TIME
C
      DRIVE(MDRV) = 0.D0
C
      DO 8 M=1,N
    8 DRIVE(MDRV) = DRIVE(MDRV)+THA(M)*FLAG(M,MDRV)
C
    9 DRIVE(MDRV) = DRVIN(MDRV)+DRIVE(MDRV)
C
   10 M3 = M3+1
C
C
C     PICK TYPE EQN TO SOLVE
C
      IF (IWAIT.EQ.0) T = T+H2
      IF (ID(M3).EQ.1) GO TO 11
      IF (ID(M3).EQ.2) GO TO 12
      IF (ID(M3).EQ.3) GO TO 13
      IF (ID(M3).EQ.4) GO TO 14
      IF (ID(M3).EQ.5) GO TO 15
      IF (ID(M3).EQ.6) GO TO 16
      WRITE (6,28)
      STOP
C
C     ***START SOLUTION***
C
C     TYPE ONE EQUATIONS
C     SOLVES  THAOUT = G*THAIN
C
   11 THA(M3) = G(M3)*DRIVE(M3)
      IWAIT = IWAIT+1
      ILAST = ILAST+1
      IF ((ILAST.EQ.N11).AND.(ICONT.EQ.NEQ)) GO TO 21
      GO TO 18
C
C     TYPE TWO EQNS
C     SOLVES THAOUT/THAIN = G/(S + P)
C
   12 THADOT(M3) = -P(M3)*THA(M3)+G(M3)*DRIVE(M3)
C
      S = RKLDE2(THA,THADOT,NR2)
      IWAIT = IWAIT+1
      IF (S-1.0) 17,18,21
C
C     TYPE THREE EQNS
C     SOLVES  THAOUT/THAIN = G*(S + Z)/(S + P)
C
   13 OMGDOT(M3) = -P(M3)*OMG(M3)+G(M3)*DRIVE(M3)
      S = RKLDE3(OMG,OMGDOT,NR3)
      THA(M3) = (Z(M3)-P(M3))*OMG(M3)+G(M3)*DRIVE(M3)
      IWAIT = IWAIT+1
      IF (S-1.0) 17,18,21
C
C     TYPE FOUR EQNS.
C     SOLVES  THAOUT/THAIN = G/(S**2 + 2*DELTA*WN*S + WN**2)
C
   14 V = CCPLX(P,Z,G,DRIVE,X2,OMG,NR4)
      THA(M3) = OMG(M3)
      IWAIT = IWAIT+1
      IF (V-1.) 17,18,21
C
C     TYPE FIVE EQNS
C     SOLVES  THAOUT = G*THAIN  FOR /G*THAIN/ < SATURATION LIMITS
C
   15 THA(M3) = G(M3)*DRIVE(M3)
      IF (THA(M3).GT.P(M3)) THA(M3)=P(M3)
      IF (THA(M3).LT.Z(M3)) THA(M3)=Z(M3)
      IWAIT = IWAIT+1
      I5LAST = I5LAST+1
      IF ((I5LAST.EQ.N55).AND.(ICONT.EQ.NEQ)) GO TO 21
      GO TO 18
C
C     TYPE SIX EQNS
C     SOLVES  THAOUT = G*THAIN  FOR CONTROL REF > BOUND DEAD SPACE
C
   16 THA(M3) = G(M3)*DRIVE(M3)
      IP = P(M3)
      IF ((CABS(THA(IP))).LT.Z(M3)) THA(M3)=0.
      IWAIT = IWAIT+1
      I6LAST = I6LAST+1
      IF ((I6LAST.EQ.N66).AND.(ICCNT.EQ.NEQ)) GO TO 21
      GO TO 18
   17 WRITE (6,29)
      STOP
   18 ICCNT = ICCNT+1
      IF (ICONT-N) 10,19,20
C
C     ONE PASS HAS BEEN MADE FOR EACH EQN. GO TO NEXT STEP INCREMENT
C
   19 NR2 = NR2+1
      NR3 = NR3+1
      NR4 = NR4+1
      N3 = 0
      ICCNT = 0
      IF (IWAIT.EQ.ICK) IWAIT=0
      GO TO 7
C
   20 WRITE (6,30)
      STOP
C
C     DATA COLLECTION POINT.  DATA IS RECORDED AT END OF COMPLETE
C     TIME STEP(DT)
C
   21 IT = IT+1
      THAOUT(IT) = THA(IOUT)
C
C     TEST FOR END OF RUN
C
      IF (T-TF) 22,22,23
C     RESET COUNTERS FOR NEXT PASS THRU EQUATIONS
   22 NR2 = 1
      NR3 = 1
      NR4 = 1
      N3 = 0
      ICCNT = 0
      IWAIT = 0
      ILAST = 0
      I5LAST = 0
      I6LAST = 0
      GO TO 7
   23 FLANT = 1.
      RETURN
C
   24 FORMAT (I2,2X,I2)
   25 FORMAT (3X,I2,I1,1X,2I2,8X,3E20.7)
   26 FORMAT (/,5H BLK ,I2,I1,2H =,2I2,6X,3E20.7)
   27 FORMAT (//,2X,'THETA CUT IS THA(',I2,')')
   28 FORMAT (40H *** EQN SWITCH CONTROL DID NOT WORK ***)
   29 FORMAT (20H INTEGRATION TROUBLE)
   30 FORMAT (58H ERROR IN INTERGATION. ATTEMPTED TO INTEG. MORE THAN N-
     1EQN)
      END
C
C         FORTRAN 4 VERSION OFRUNGE-KUTTA-GILL ROUTINE
C          X,XDOT,T,DT, ARE IN DOUBLE PRECISION
C                    MAX N=25
C
      FUNCTION RKLDEQ(N,X,XDOT,T,DT,NT)
      REAL*8 X,XDOT,T,DT,Q,H1,H2,H3,H6
      DIMENSION X(1),XDOT(1),Q(25)
C
      NT = NT + 1
      GO TO (1,2,3,4),NT
    1 H1 = DT
      H2 = H1*0.5D0
      H3 = H1*2.0D0
      H6 = H1/6.0D0
      DO 11 J = 1,N
   11 Q(J) = 0.D0
      A = 0.5D0
      T = T + H2
      GO TO 5
C
    2 A = 0.2928932188134525
      GO TO 5
C
    3 A = 1.7071067811865475
      T = T + H2
      GO TO 5
C
    4 DO 41 I = 1,N
   41 X(I) = X(I) + H6*XDOT(I) - Q(I)/3.D0
      NT = 0
      RKLDEQ = 2.
      GO TO 6
C
    5 DO 51 L = 1,N
      X(L) = X(L) + A*(DT*XDOT(L)-Q(L))
   51 Q(L) = H3*A*XDOT(L) + (1.D0 - 3.D0*A)*Q(L)
      RKLDEQ = 1.
C
    6 RETURN
      END
C
C         FORTRAN 4 VERSION OFRUNGE-KUTTA-GILL ROUTINE
```

```
C          X,XDOT,T,DT,  ARE IN DOUBLE FRECISION
C                        MAX N=25
C
      FUNCTION RKLDE2 (X,XDOT,NR2)
      DIMENSION X(4), XDOT(4), Q(25)
      DIMENSION THAOUT(3001), XDATA(3001)
      REAL *8THAOUT,XDATA
      REAL *8X,XDOT,T,DT,Q,H1,H2,H3,H6
      REAL *8TF
      COMMON T,DT,TF,THAOUT,XDATA,M3,ICONT,NEQ,ISKIP,ITF
C
      GO TO (1,2,3,4), NR2
    1 H1 = DT
      H2 = H1*0.5D0
      H3 = H1*2.0D0
      H6 = H1/6.0D0
      Q(M3) = 0.D0
      A = 0.5D0
      GO TO 5
C
    2 A = 0.2928932188134525
      GO TO 5
C
    3 A = 1.7071067811865475
      GO TO 5
C
    4 X(M3) = X(M3)+H6*XDOT(M3)-Q(M3)/3.D0
      RKLDE2 = 1.
      IF (ICONT.EQ.NEQ) RKLDE2=2.
      GO TO 6
C
    5 X(M3) = X(M3)+A*(DT*XDOT(M3)-Q(M3))
      Q(M3) = H3*A*XDOT(M3)+(1.D0-3.D0*A)*Q(M3)
      RKLDE2 = 1.
C
    6 RETURN
      END
C
      FUNCTION RKLDE3 (X,XDOT,NR3)
      DIMENSION X(4), XDOT(4), Q(25)
      DIMENSION THAOUT(3001), XDATA(3001)
      REAL *8THAOUT,XDATA
      REAL *8X,XDOT,T,DT,Q,H1,H2,H3,H6
      REAL *8TF
      COMMON T,DT,TF,THAOUT,XDATA,M3,ICONT,NEQ,ISKIP,ITF
C
      GO TO (1,2,3,4), NR3
    1 H1 = DT
```

```
      H2 = H1*0.5D0
      H3 = H1*2.0D0
      H6 = H1/6.0D0
      Q(M3) = 0.D0
      A = 0.5D0
      GO TO 5
C
    2 A = 0.2928932188134525
      GO TO 5
C
    3 A = 1.7071067811865475
      GO TO 5
C
    4 X(M3) = X(M3)+H6*XDOT(M3)-Q(M3)/3.D0
      RKLDE3 = 1.
      IF (ICONT.EQ.NEQ) RKLDE3=2.
      GO TO 6
C
    5 X(M3) = X(M3)+A*(DT*XDOT(M3)-Q(M3))
      Q(M3) = H3*A*XDOT(M3)+(1.D0-3.D0*A)*Q(M3)
      RKLDE3 = 1.
    6 RETURN
      END
C
C                          FUNCTION CMPLX
C
C     THIS FUNCTION IS USED TO COMPUTE THE RESPONSE OF A COMPLEX
C     TRANSFER FUNCTION  (G/(S**2 + 2*E*W*S + W**2))
C
      FUNCTION CCPLX (P,Z,G,DRIVE,X2,OMG,NR4)
      DIMENSION OMG(1), P(1), Z(1), G(1), DRIVE(1), OMGDOT(1), X2(25), X
     12DOT(25)
      DIMENSION THAOUT(3001), XDATA(3001)
      REAL *8THAOUT,XDATA
      REAL *8OMG,OMGDOT,DRIVE,X2,X2DOT,T,DT
      REAL *8TF
      COMMON T,DT,TF,THAOUT,XDATA,M3,ICONT,NEQ,ISKIP,ITF
C
      OMGDOT(M3) = X2(M3)
C
      SS = RKLDE3(OMG,OMGDOT,NR4)
C
      X2DOT(M3) = -2.*P(M3)*Z(M3)*X2(M3)-Z(M3)**2*OMG(M3)+G(M3)*DRIVE(M3
     1)
C
      SSS = RKLDE4(X2,X2DOT,NR4)
      CCPLX = SSS
      RETURN
      END
C
C     THIS FUNCTION WORKS ONLY WITH COMPLEX ROUTINE
C
      FUNCTION RKLDE4 (X,XDOT,NR4)
      DIMENSION X(1), XDOT(1), QC(25)
      DIMENSION THAOUT(3001), XDATA(3001)
      REAL *8THAOUT,XDATA
      REAL *8X,XDOT,T,DT,QC,H1,H2,H3,H6
      REAL *8TF
      COMMON T,DT,TF,THAOUT,XDATA,M3,ICONT,NEQ,ISKIP,ITF
C
      GO TO (1,2,3,4), NR4
    1 H1 = DT
      H2 = H1*0.5D0
      H3 = H1*2.0D0
      H6 = H1/6.0D0
      QC(M3) = 0.D0
      A = 0.5D0
      GO TO 5
C
    2 A = 0.29289321881345245
      GO TO 5
C
    3 A = 1.7071067811865475
      GO TO 5
C
    4 X(M3) = X(M3)+H6*XDOT(M3)-QC(M3)/3.D0
      RKLDE4 = 1.
      IF (ICONT.EQ.NEQ) RKLDE4=2.
      GO TO 6
C
    5 X(M3) = X(M3)+A*(DT*XDOT(M3)-QC(M3))
      QC(M3) = H3*A*XDOT(M3)+(1.D0-3.D0*A)*QC(M3)
      RKLDE4 = 1.
C
    6 RETURN
      END
C
      SUBROUTINE PIC (XDATA,THAOUT,DT,NPPLT,IDP)
C
C     THIS ROUTINE PLOTS THE SYSTEM AND DESIRED RESPONSES.
C     REQUIRES APPROXIMATELY 28K FOR STORAGE AND IN--OUT BUFFERS
C
      DIMENSION XX(900), YY(900), WW(900)
      DIMENSION TX(4), TY(4)
      REAL *8XDATA(501),THAOUT(501),TITLE(12),DT
      REAL *8LABC/'        '/
      REAL *4LABA/' A  '/,LABC/'  D '/
      READ (5,2) TITLE
C
C     TITLE IS ON TWO CARDS AND MUST HAVE YOUR ID
C     AND GRAPH TITLE IN COL 1-48.
C
      T = 0.
      TSTEP = 5.0*DT
      J = 0
      BIGX = 0.
      BIGY = 0.
      SMLX = 0.
      SMLY = 0.
C
      DO 1 I=1,IDP,5
      J = J+1
      XX(J) = T
      YY(J) = XDATA(I)
      WW(J) = THAOUT(I)
      X = T
      XD = YY(J)
      TH = WW(J)
      YMAX = AMAX1(XD,TH)
      YMIN = AMIN1(XD,TH)
      XMAX = AMAX1(BIGX,X)
      IF (BIGX.LT.X) BIGX=X
      IF (BIGY.LT.YMAX) BIGY=YMAX
      IF (SMLY.GT.YMIN) SMLY=YMIN
      T = T+TSTEP
    1 CONTINUE
C
C
      TX(1) = 0.
      TX(2) = 0.
      TX(3) = SMLX
      TX(4) = BIGX
      TY(1) = BIGY
      TY(2) = SMLY
      TY(3) = 0.
      TY(4) = 0.
C
C     PLOT SYSTEM RESPONSE
C     SYMBL 'A' IS ACTUAL RESPONSE, SYMBL 'D' IS DESIRED DATA RESPONSE
C
      CALL DRAW (4,TX,TY,1,1,LABC,TITLE,0,0,0,0,0,0,8,8,0,L)
      CALL DRAW (NPPLT,XX,WW,2,0,LABA,TITLE,0,0,0,0,0,0,8,8,0,L)
C
C     PLOT DESIRED RESPONSE
```

```
C
      CALL DRAW (NPPLT,XX,YY,3,0,LABC,TITLE,0,0,0,0,0,0,8,8,0,L)
      IF (L.NE.0) WRITE (6,3) L
      RETURN
C
    2 FORMAT (6A8)
    3 FORMAT (//,'  GRAPH NOT COMPLETED. OUTPUT CODE = ',I2)
      END
C
      SUBROUTINE PPLT (XDATA,THAOUT,DT,NPPLT,IDP)
C
C     THIS ROUTINE PRODUCES THE PRINTER PLOTS.
C
      DIMENSION XX(900), YY(900), WW(900)
      DIMENSION TX(4), TY(4)
      REAL *8XDATA(3001),THAOUT(3001),DT
C
C     CHANGE VARIABLES TO REAL*4 AND SET UP PLOT POINTS.
C
C     CLEAR REGISTERS AND SET AXIS. REG. PROBABLY DO NOT HAVE TO CLEAR N
C     CHECK ON LATER......
C
      DO 1 I=1,900
      XX(I) = 0.
      YY(I) = 0.
    1 WW(I) = 0.
C
      T = 0.
      TSTEP = 5.0*DT
      J = 0
      BIGX = 0.
      BIGY = 0.
      SMLX = 0.
      SMLY = 0.
C
      DO 2 I=1,IDP,5
      J = J+1
      XX(J) = T
      YY(J) = XDATA(I)
      WW(J) = THAOUT(I)
C
      X = T
      XD = YY(J)
      TH = WW(J)
      YMAX = AMAX1(XD,TH)
      YMIN = AMIN1(XD,TH)
      XMAX = AMAX1(BIGX,X)
      IF (BIGX.LT.X) BIGX=X
      IF (BIGY.LT.YMAX) BIGY=YMAX
      IF (SMLY.GT.YMIN) SMLY=YMIN
      T = T+TSTEP
    2 CONTINUE
C
      TX(1) = 0.
      TX(2) = 0.
      TX(3) = SMLX
      TX(4) = BIGX
      TY(1) = BIGY
      TY(2) = SMLY
      TY(3) = 0.
      TY(4) = 0.
      WRITE (6,3) BIGX,BIGY,SMLY
C
C     PLOT SYSTEM RESPONSE. "*" IS DESIRED RESPONSE. "+" IS ACTUAL
C     RESPONSE.  "*" ONLY INDICATE COINCIDENCE OF CURVES.
C
      CALL PLOTP (TX,TY,4,1)
      CALL PLOTP (XX,WW,NPPLT,2)
      CALL PLOTP (XX,YY,NPPLT,3)
      WRITE (6,4)
      IF (IDP.EQ.4500) WRITE (6,5)
      RETURN
C
    3 FORMAT (2X,'BIGX= ',E15.7,2X,'BIGY= ',E15.7,2X,'SMLY= ',E15.7)
    4 FORMAT (//,2X,'SYSTEM RESPONSE FOR PROBLEM _____')
    5 FORMAT (//,2X,'STOP AT 500 GRAPH POINTS')
      END
C     .................................................................
C
C     SUBROUTINE BOXPLX                 (CATEGORY H0)
C
C     PURPOSE
C
C        BOXPLX IS A SUBROUTINE USED TO SOLVE THE PROBLEM OF LOCATING
C        A MINIMUM (OR MAXIMUM) OF AN ARBITRARY OBJECTIVE FUNCTION
C        SUBJECT TO ARBITRARY EXPLICIT AND/OR IMPLICIT CONSTRAINTS BY
C        THE COMPLEX METHOD OF M.J. BOX.  EXPLICIT CONSTRAINTS ARE
C        DEFINED AS UPPER AND LOWER BOUNDS ON THE INDEPENDENT VARIABLES.
C        IMPLICIT CONSTRAINTS MAY BE ARBITRARY FUNCTIONS OF THE VAR-
C        IABLES.  TWO FUNCTION SUBPROGRAMS TO EVALUATE THE OBJECTIVE
C        FUNCTION AND IMPLICIT CONSTRAINTS, RESPECTIVELY, MUST BE
C        SUPPLIED BY THE USER (SEE EXAMPLE BELOW).  BOXPLX ALSO HAS
C        THE OPTION TO PERFORM INTEGER PROGRAMMING, WHERE THE VALUES
C        OF THE INDEPENDENT VARIABLES ARE RESTRICTED TO INTEGERS.
C
C     USAGE
```

```
C                                                                       BXPX0200
C      CALL BOXPLX (NV,NAV,NPR,NTA,R,XS,IP,XU,XL,YMN,IER)               BXPX0210
C                                                                       BXPX0220
C                                                                       BXPX0230
C   DESCRIPTION OF PARAMETERS                                           BXPX0240
C                                                                       BXPX0250
C      NV    AN INTEGER INPUT DEFINING THE NUMBER OF INDEPENDENT        BXPX0260
C            VARIABLES OF THE OBJECTIVE FUNCTION TO BE MINIMIZED.       BXPX0270
C            NOTE:  MAXIMUM NV + NAV IS PRESENTLY 50.  MAXIMUM NV IS    BXPX0280
C            25.  IF THESE LIMITS MUST BE EXCEEDED, PUNCH A SOURCE      BXPX0290
C            DECK IN THE USUAL MANNER, AND CHANGE THE DIMENSION         BXPX0300
C            STATEMENTS.                                                BXPX0310
C                                                                       BXPX0320
C      NAV   AN INTEGER INPUT DEFINING THE NUMBER OF AUXILIARY VAR-     BXPX0330
C            IABLES THE USER WISHES TO DEFINE FOR HIS OWN CONVENIENCE.  BXPX0340
C            TYPICALLY HE MAY WISH TO DEFINE THE VALUE OF EACH IMPLICITBXPX0350
C            CONSTRAINT FUNCTION AS AN AUXILIARY VARIABLE.  IF THIS     BXPX0360
C            IS DONE, THE OPTIONAL OUTPUT FEATURE OF BOXPLX CAN BE      BXPX0370
C            USED TO OBSERVE THE VALUES OF THOSE CONSTRAINTS AS THE     BXPX0380
C            SOLUTION PROGRESSES.  AUXILIARY VARIABLES, IF USED,        BXPX0390
C            SHOULD BE EVALUATED IN FUNCTION KE (DEFINED BELOW).        BXPX0400
C            NAV MAY BE ZERO.                                           BXPX0410
C                                                                       BXPX0420
C      NPR   INPUT INTEGER CONTROLLING THE FREQUENCY OF OUTPUT DESIRED  BXPX0430
C            FOR DIAGNOSTIC PURPOSES.  IF NPR .LE. 0, NO OUTPUT WILL BEBXPX0440
C            PRODUCED BY BOXPLX.  OTHERWISE, THE CURRENT COMPLEX OF     BXPX0450
C            K= 2*NV VERTICES AND THEIR CENTROID WILL BE OUTPUT AFTER   BXPX0460
C            EACH NPR PERMISSIBLE TRIALS.  THE NUMBER OF TOTAL TRIALS,  BXPX0470
C            NUMBER OF FEASIBLE TRIALS, NUMBER OF FUNCTION EVALUATIONS  BXPX0480
C            AND NUMBER OF IMPLICIT CONSTRAINT EVALUATIONS ARE IN-      BXPX0490
C            CLUDED IN THE OUTPUT.                                      BXPX0500
C            ADDITIONALLY, (WHEN NPR .GT. 0) THE SAME INFORMATION       BXPX0510
C            WILL BE OUTPUT:                                            BXPX0520
C                                                                       BXPX0530
C              1) IF THE INITIAL POINT IS NOT FEASIBLE,                 BXPX0540
C              2) AFTER THE FIRST COMPLETE COMPLEX IS GENERATED,        BXPX0550
C              3) IF A FEASIBLE VERTEX CANNOT BE FOUND AT SOME TRIAL,   BXPX0560
C              4) IF THE OBJECTIVE VALUE OF A VERTEX CANNOT BE MADE     BXPX0570
C                 NO-LONGER-WORST.                                      BXPX0580
C              5) IF THE LIMIT ON TRIALS (NTA) IS REACHED AND,          BXPX0590
C              6) WHEN THE OBJECTIVE FUNCTION HAS BEEN UNCHANGED FOR    BXPX0600
C                 2*NV TRIALS, INDICATING A LOCAL MINIMUM HAS BEEN      BXPX0610
C                 FOUND.                                                BXPX0620
C                                                                       BXPX0630
C            IF THE USER WISHES TO TRACE THE PROGRESS OF A SOLUTION,    BXPX0640
C            A CHOICE OF NPR = 25, 50 OR 100 IS RECOMMENDED.            BXPX0650
C                                                                       BXPX0660
C      NTA   INTEGER INPUT OF LIMIT ON THE NUMBER OF TRIALS ALLOWED     BXPX0670
```

```
C           IN THE CALCULATION.  IF THE USER INPUTS NTA .LE. 0, A      BXPX0680
C           DEFAULT VALUE OF 2000 IS USED.  WHEN THIS LIMIT IS REACHEDBXPX0690
C           CONTROL RETURNS TO THE CALLING PROGRAM WITH THE BEST       BXPX0700
C           ATTAINED OBJECTIVE FUNCTION VALUE IN YMN, AND THE BEST     BXPX0710
C           ATTAINED SOLUTION POINT IN XS.                             BXPX0720
C                                                                      BXPX0730
C     R     A REAL NUMBER INPUT TO DEFINE THE FIRST RANDOM NUMBER      BXPX0740
C           USED IN DEVELOPING THE INITIAL COMPLEX OF 2*NV VERTICIES.  BXPX0750
C           (0. .GT. R .LT. 1.) IF R IS NOT WITHIN THESE BOUNDS,       BXPX0760
C           IT WILL BE REPLACED BY 1./3. .                             BXPX0770
C                                                                      BXPX0780
C     XS    INPUT REAL ARRAY DIMENSIONED AT LEAST NV+NAV. THE FIRST    BXPX0790
C           NV MUST CONTAIN A FEASIBLE ORIGIN FOR STARTING THE CAL-    BXPX0800
C           CULATION.  THE LAST NAV NEED NOT BE INITIALIZED.  UPON     BXPX0810
C           RETURN FROM BOXPLX, THE FIRST NV ELEMENTS OF THE ARRAY     BXPX0820
C           CONTAIN THE COORDINATES OF THE MINIMUM OBJECTIVE FUNCTION,BXPX0830
C           AND THE REMAINING NAV (NAV .GE. 0) CONTAIN THE VALUES OF   BXPX0840
C           THE CORRESPONDING AUXILIARY VARIABLES.                     BXPX0850
C                                                                      BXPX0860
C     IP    INTEGER INPUT FOR OPTIONAL INTEGER PROGRAMMING.  IF IP=1,  BXPX0870
C           THE VALUES OF THE INDEPENDENT VARIABLES WILL BE REPLACED   BXPX0880
C           WITH INTEGER VALUES (STILL STORED AS REAL*4).              BXPX0890
C                                                                      BXPX0900
C     XU    A REAL ARRAY DIMENSIONED AT LEAST NV INPUTTING THE UPPER   BXPX0910
C           BOUND ON EACH INDEPENDENT VARIABLE, (EACH EXPLICIT CON-    BXPX0920
C           STRAINT).  INPUT VALUES ARE SLIGHTLY ALTERED BY BOXPLX.    BXPX0930
C                                                                      BXPX0940
C     XL    A REAL ARRAY DIMENSIONED AT LEAST NV INPUTTING THE LOWER   BXPX0950
C           BOUND ON EACH INDEPENDENT VARIABLE, (EACH EXPLICIT CON-    BXPX0960
C           STRAINT).  NOTE:  FOR BOTH XU AND XL CHOOSE REASONABLE     BXPX0970
C           VALUES IF NONE ARE GIVEN, NOT VALUES WHICH ARE MAGNITUDES  BXPX0980
C           ABOVE OR BELOW THE EXPECTED SOLUTION.  INPUT VALUES ARE    BXPX0990
C           SLIGHTLY ALTERED BY BOXPLX.                                BXPX1000
C                                                                      BXPX1010
C     YMN   THIS OUTPUT IS THE VALUE (REAL*4) OF THE OBJECTIVE FUNC-   BXPX1020
C           TION, CORRESPONDING TO THE SOLUTION POINT OUTPUT IN XS.    BXPX1030
C                                                                      BXPX1040
C     IER   INTEGER ERROR RETURN.  TO BE INTERROGATED UPON RETURN      BXPX1050
C           FROM BOXPLX.  IER WILL BE ONE OF THE FOLLOWING:            BXPX1060
C                                                                      BXPX1070
C              =-1  CANNOT FIND FEASIBLE VERTEX OR FEASIBLE CENTROID   BXPX1080
C                   AT THE START OR A RESTART (SEE 'METHOD' BELOW).    BXPX1090
C              =0   FUNCTION VALUE UNCHANGED FOR 'N' TRIALS.  (WHERE   BXPX1100
C                   N=6*NV+10)  THIS IS THE NORMAL RETURN PARAMETER.   BXPX1110
C              =1   CANNOT DEVELOP FEASIBLE VERTEX.                    BXPX1120
C              =2   CANNOT DEVELOP A NO-LONGER-WORST VERTEX.           BXPX1130
C              =3   LIMIT ON TRIALS REACHED.  (NTA EXCEEDED)           BXPX1140
C              NOTE:  VALID RESULTS MAY BE RETURNED IN ANY OF THE      BXPX1150
```

```
C                    ABOVE CASES.                                        BXPX1160
C                                                                        BXPX1170
C     EXAMPLE OF USAGE                                                   BXPX1180
C                                                                        BXPX1190
C        THIS EXAMPLE MINIMIZES THE OBJECTIVE FUNCTION SHOWN IN THE      BXPX1200
C        EXTERNAL FUNCTION FE(X).  THERE ARE TWO INDEPENDENT VAR-        BXPX1210
C        IABLES X(1) & X(2), AND TWO IMPLICIT CONSTRAINT FUNCTIONS       BXPX1220
C        X(3) & X(4) WHICH ARE EVALUATED AS AUXILIARY VARIABLES (SEE     BXPX1230
C        EXTERNAL FUNCTION KE(X) ).                                      BXPX1240
C                                                                        BXPX1250
C     DIMENSION  XS(4),XU(2),XL(2)                                       BXPX1260
CC                                                                       BXPX1270
CC   STARTING GUESS                                                      BXPX1280
C     XS(1) = 1.0                                                        BXPX1290
C     XS(2) = 0.5                                                        BXPX1300
CC   UPPER LIMITS                                                        BXPX1310
C     XU(1) = 6.0                                                        BXPX1320
C     XU(2) = 6.0                                                        BXPX1330
CC   LOWER LIMITS                                                        BXPX1340
C     XL(1) = 0.0                                                        BXPX1350
C     XL(2) = 0.0                                                        BXPX1360
CC                                                                       BXPX1370
C     R = 9./13.                                                         BXPX1380
C     NTA = 5000                                                         BXPX1390
C     NPR = 50                                                           BXPX1400
C     NAV = 2                                                            BXPX1410
C     NV = 2                                                             BXPX1420
C     IP = 0                                                             BXPX1430
CC                                                                       BXPX1440
C      CALL BCXPLX (NV,NAV,NPR,NTA,R,XS,IP,XU,XL,YMN,IER)                BXPX1450
C     WRITE(6,1) ((XS(I),I=1,4),YMN,IER)                                 BXPX1460
C   1 FORMAT (////,'   THE POINT IS LOCATED AT (XS(I)=) ',4(E13.7,5X),   BXPX1470
C    1//,'   AND THE FUNCTION VALUE IS ',E13.7,' IER = ',I5)             BXPX1480
CC                                                                       BXPX1490
C     STOP                                                               BXPX1500
C     END                                                                BXPX1510
CC                                                                       BXPX1520
CC                                                                       BXPX1530
C     FUNCTION  KE(X)                                                    BXPX1540
CC   EVALUATE CONSTRAINTS.  SET KE=0 IF NO IMPLICIT CONSTRAINT IS        BXPX1550
CC   VIOLATED, OR SET KE=1 IF ANY.IMPLICIT CONSTRAINT IS VIOLATED.       BXPX1560
C     DIMENSION X(4)                                                     BXPX1570
C     X1 = X(1)                                                          BXPX1580
C     X2 = X(2)                                                          BXPX1590
C     KE = 0                                                             BXPX1600
C     X(3) = X1 + 1.732051*X2                                            BXPX1610
C     IF (X(3) .LT. 0. .OR. X(3) .GT. 6.) GO TO 1                        BXPX1620
C     X(4) = X1/1.732051 -X2                                             BXPX1630
```

```
C      IF (X(4) .GE. 0.)  RETURN                                         BXPX1640
CC                                                                       BXPX1650
C    1 KE = 1                                                            BXPX1660
C      RETURN                                                            BXPX1670
C      END                                                               BXPX1680
CC                                                                       BXPX1690
CC                                                                       BXPX1700
C      FUNCTION  FE(X)                                                   BXPX1710
C      DIMENSION X(4)                                                    BXPX1720
CC                                                                       BXPX1730
CC   THIS IS THE OBJECTIVE FUNCTION.                                     BXPX1740
C      FE= -(X(2)**3 *(9.-(X(1)-3.)**2)/(46.76538))                      BXPX1750
C      RETURN                                                            BXPX1760
C      END                                                               BXPX1770
C                                                                        BXPX1780
C      METHOD                                                            BXPX1790
C                                                                        BXPX1800
C         THE COMPLEX METHOD IS AN EXTENSION AND ADAPTION OF THE SIM-    BXPX1810
C         PLEX METHOD OF LINEAR PROGRAMMING.  STARTING WITH ANY ONE      BXPX1820
C         FEASIBLE POINT IN N-DIMENSION SPACE A "COMPLEX" OF 2*N         BXPX1830
C         VERTICES IS CONSTRUCTED BY SELECTING RANDOM POINTS WITHIN THE  BXPX1840
C         FEASIBLE REGION.  FOR THIS PURPOSE N COORDINATES ARE FIRST     BXPX1850
C         RANDOMLY CHOSEN WITHIN THE SPACE BOUNDED BY EXPLICIT CON-      BXPX1860
C         STRAINTS.  THIS DEFINES A TRIAL INITIAL VERTEX.  IT IS THEN    BXPX1870
C         CHECKED FOR POSSIBLE VIOLATION OF IMPLICIT CONSTRAINTS.  IF    BXPX1880
C         ONE OR MORE ARE VIOLATED, THE TRIAL INITIAL VERTEX IS DISPLACEDBXPX1890
C         HALF OF ITS DISTANCE FROM THE CENTROID OF PREVIOUSLY SELECTED  BXPX1900
C         INITIAL VERTICES.  IF NECESSARY THIS DISPLACEMENT PROCESS IS   BXPX1910
C         REPEATED UNTIL THE VERTEX HAS BECOME FEASIBLE.  IF THIS FAILS  BXPX1920
C         TO HAPPEN AFTER 5*N+10 DISPLACEMENTS, THE SOLUTION IS ABAND-   BXPX1930
C         ONED.  AFTER EACH VERTEX IS ADDED TO THE COMPLEX, THE CURRENT  BXPX1940
C         CENTROID IS CHECKED FOR FEASIBILITY.  IF IT IS INFEASIBLE,     BXPX1950
C         THE LAST TRIAL VERTEX IS ABANDONED AND AN EFFORT TO GENERATE   BXPX1960
C         AN ALTERNATIVE TRIAL VERTEX IS MADE.  IF 5*N+10 VERTICES ARE   BXPX1970
C         ABANDONED CONSECUTIVELY, THE SOLUTION IS TERMINATED.           BXPX1980
C                                                                        BXPX1990
C         IF AN INITIAL COMPLEX IS ESTABLISHED, THE BASIC COMPUTATION    BXPX2000
C         LOOP IS INITIATED.  THESE INSTRUCTIONS FIND THE CURRENT WORST  BXPX2010
C         VERTEX, THAT IS, THE VERTEX WITH THE LARGEST CORRESPONDING     BXPX2020
C         VALUE FOR THE OBJECTIVE FUNCTION, AND REPLACE THAT VERTEX BY   BXPX2030
C         ITS OVER-REFLECTION THROUGH THE CENTROID OF ALL OTHER VERTICES.BXPX2040
C         (IF THE VERTEX TO BE REPLACED IS CONSIDERED AS A VECTOR IN     BXPX2050
C         N-SPACE, ITS OVER-REFLECTION IS OPPOSITE IN DIRECTION, IN-     BXPX2060
C         CREASED IN LENGTH BY THE FACTOR 1.3, AND COLLINEAR WITH        BXPX2070
C         THE REPLACED VERTEX AND CENTROID OF ALL OTHER VERTICES.)       BXPX2080
C                                                                        BXPX2090
C         WHEN AN OVER-REFLECTION IS NOT FEASIBLE OR REMAINS WORST, IT   BXPX2100
C         IS CONSIDERED NOT-PERMISSIBLE AND IS DISPLACED HALFWAY TOWARD  BXPX2110
```

```
C        THE CENTROID.  AFTER FOUR SUCH ATTEMPTS ARE MADE UNSUCCESSFULLYBXPX2120
C        EVERY FIFTH ATTEMPT IS MADE BY REFLECTING THE OFFENDING VERTEX BXPX2130
C        THROUGH THE PRESENT BEST VERTEX, INSTEAD OF THROUGH THE CEN-   BXPX2140
C        TROID.  IF 5*N+10 DISPLACEMENTS AND OVER-REFLECTIONS OCCUR     BXPX2150
C        WITHOUT A SUCCESSFUL (PERMISSIBLE) RESULT, THE CURRENT BEST    BXPX2160
C        VERTEX IS TAKEN AS AN INITIAL FEASIBLE POINT FOR A RESTART     BXPX2170
C        RUN OF THE COMPLETE PROCESS.  RESTARTING IS ALSO UNDERTAKEN    BXPX2180
C        WHEN 6*NV+10 CONSECUTIVE TRIALS HAVE BEEN MADE WITH NO SIGNIF- BXPX2190
C        ICANT CHANGE IN THE VALUE OF THE OBJECTIVE FUNCTION.  IN ALL   BXPX2200
C        CASES, RESTARTING IS INHIBITED IF THE LAST RESTART DID NOT     BXPX2210
C        PRODUCE A SIGNIFICANT IMPROVEMENT IN THE MINIMUM ATTAINED.     BXPX2220
C                                                                       BXPX2230
C        IT IS RECOMMENDED THAT THE USER READ THE REFERENCE FOR         BXPX2240
C        FURTHER USEFUL INFORMATION.  IT SHOULD BE NOTED THAT THE       BXPX2250
C        ALGORITHM DEFINED THERE HAS BEEN ALTERED TO FIND THE           BXPX2260
C        CONSTRAINED MINIMUM, RATHER THAN THE MAXIMUM.                  BXPX2270
C                                                                       BXPX2280
C                                                                       BXPX2290
C                                                                       BXPX2300
C     REMARKS                                                           BXPX2310
C                                                                       BXPX2320
C        THE INTEGER PROGRAMMING OPTION WAS ADDED TO THIS PROGRAM       BXPX2330
C        AS SUGGESTED IN REFERENCE (2).  A MIXED INTEGER/CONTINUOUS     BXPX2340
C        VARIABLE VERSION OF BOXPLX WOULD BE EASY TO CREATE BY DE-      BXPX2350
C        CLARING "IP" TO BE AN ARRAY OF NV CONTROL VARIABLES WHERE IP   BXPX2360
C        (I)=1 WOULD INDICATE THAT THE I-TH VARIABLE IS TO BE CONFINED  BXPX2370
C        TO INTEGER VALUES.  EACH STATEMENT OF THE FORM 'IF (IP .EQ.    BXPX2380
C        1)' ETC. WOULD THEN NEED TO BE ALTERED TO 'IF (IP(I) .EQ. 1)'  BXPX2390
C        ETC., WHERE THE SUBSCRIPT IS APPROPRIATELY CHOSEN.  NORMALLY,  BXPX2400
C        XU AND XL VALUES ARE ALTERED TO BE AN EPSILON 'WITHIN' ACTUAL  BXPX2410
C        VALUES DECLARED BY THE USER.  THIS ADJUSTMENT IS NOT MADE      BXPX2420
C        WHEN IP=1.                                                     BXPX2430
C                                                                       BXPX2440
C        NOTE:  NO NON-LINEAR PROGRAMMING ALGORITHM CAN GUARANTEE THAT  BXPX2450
C        THE ANSWER FOUND IS THE GLOBAL MINIMUM, RATHER THAN JUST A     BXPX2460
C        LOCAL MINIMUM.  HOWEVER, ACCORDING TO REF.2, THE COMPLEX       BXPX2470
C        METHOD HAS AN ADVANTAGE IN THAT IT TENDS TO FIND THE GLOBAL    BXPX2480
C        MINIMUM MORE FREQUENTLY THAN MANY OTHER NON-LINEAR PROGRAM-    BXPX2490
C        MING ALGORITHMS.                                               BXPX2500
C                                                                       BXPX2510
C        IT SHOULD BE NOTED THAT THE AUXILIARY VARIABLE FEATURE CAN     BXPX2520
C        ALSO BE USED TO DEAL WITH PROBLEMS CONTAINING EQUALITY CON-    BXPX2530
C        STRAINTS.  ANY EQUALITY CONSTRAINT IMPLIES THAT A GIVEN VAR-   BXPX2540
C        IABLE IS NOT TRULY INDEPENDENT.  THEREFORE, IN GENERAL, ONE    BXPX2550
C        VARIABLE INVOLVED IN AN EQUALITY CONSTRAINT CAN BE RENUMBERED  BXPX2560
C        FROM THE SET OF NV INDEPENDENT VARIABLES AND ADDED TO THE SET  BXPX2570
C        OF NAV AUXILIARY VARIABLES.  THIS USUALLY INVOLVES RENUMBERING BXPX2580
C        THE INDEPENDENT VARIABLES OF THE GIVEN PROBLEM.                BXPX2590
```

```
C
C     SUBROUTINES AND FUNCTIONS REQUIRED
C
C        SUBROUTINE 'BOUT' AND FUNCTION 'FBV' ARE INTEGRAL PARTS OF
C        THE BOXPLX PACKAGE.
C
C        TWO FUNCTIONS MUST BE SUPPLIED BY THE USER.  THE FIRST, KE(X),
C        IS USED TO EVALUATE THE IMPLICIT CONSTRAINTS.  SET KE=0 AT THE
C        BEGINING OF THE FUNCTION, THEN EVALUATE THE IMPLICIT CON-
C        STRAINTS.  IN THE EXAMPLE ABOVE, THE FIRST CONSTRAINT, X(3),
C        MUST BE WITHIN THE RANGE (0. .LE. X(3) .LE. 6.).  THE SECOND
C        CONSTRAINT X(4), MUST BE .GE. 0. .  IF EITHER CONSTRAINT IS
C        NOT WITHIN THESE BOUNDS, CONTROL IS TRANSFERRED TO STATEMENT 1,
C        AND KE IS SET TO "1" AND CONTROL IS RETURNED TO BOXPLX.
C
C        THE SECOND FUNCTION THE USER MUST PROVIDE EVALUATES THE OB-
C        JECTIVE FUNCTION.  IT IS CALLED FE(X) AS SHOWN IN THE EXAM-
C        PLE ABOVE, AND FE MUST BE SET TO THE VALUE OF THE OBJECTIVE
C        FUNCTION CORRESPONDING TO CURRENT VALUES OF THE NV INDEPENDENT
C        VARIABLES IN ARRAY 'X'.
C
C     REFERENCES
C
C        BOX, M. J., "A NEW METHOD OF CONSTRAINED OPTIMIZATION AND A
C        COMPARISON WITH OTHER METHODS", COMPUTER JOURNAL, 8 APR. '65,
C        PP. 45-52.
C
C        BEVERIDGE G., AND SCHECHTER R., "OPTIMIZATION: THEORY AND
C        PRACTICE", MCGRAW-HILL, 1970.
C
C     PROGRAMMER
C
C        R.R. HILLEARY 1/1966.
C        REVISED FOR SYSTEM 360 4/1967
C        CORRECTED  1/1969
C        REVISED/EXTENDED BY L.NOLAN/R.HILLEARY  2/1975
C
C
C     ..................................................................
C
C
      SUBROUTINE BOXPLX (NV,NAV,NPR,NTZ,RZ,XS,IP,EU,BL,YMN,IER)
C
      DIMENSION V(50,50), FUN(50), SUM(25), CEN(25), XS(NV), EU(NV), BL(
     1NV)
C
      KV = 5
      EF = 1.E-6
      NTA = 2000
      IF (NTZ.GT.0) NTA = NTZ
      F = RZ
      IF (R.LE.0..OR.R.GE.1.) R=1./3.
      NVT = NV+NAV
C
C           TOTAL VARS, EXPLICIT PLUS IMPLICIT
      NT = 0
C           CURRENT TRIAL NO.
      NFT = 0
C           CURRENT NO. OF PERMISSIBLE TRIALS
      NTFS = 0
C           CURRENT NO. OF TIMES F HAS BEEN ALMOST UNCHANGED
C
C              CHECK FEASIBILITY OF START POINT
C
      DO 4 I=1,NV
      VT = XS(I)
      IF (BL(I).LE.VT) GO TO 1
      II = -I
      VT = BL(I)
      GO TO 2
    1 IF (BU(I).GE.VT) GO TO 3
      II = I
      VT = BU(I)
    2 IF (NPR.GT.0) WRITE (6,49) II
    3 V(I,1) = VT
      CEN(I) = VT
      IF (IP.EQ.1) GO TO 4
      BL(I) = BL(I)+AMAX1(EP,EP*ABS(BL(I)))
      BL(I) = BU(I)-AMAX1(EP,EP*ABS(BU(I)))
    4 SLM(I) = VT
C
C
      NCE = 1
C           NUMBER OF CONSTRAINT EVALUATIONS
      I = 1
      IF (KE(V(1,1)).EQ.0) GO TO 5
      IF (NPR.LE.0) GO TO 12
      WRITE (6,50)
      GO TO 12
    5 NFE = 1
C
C    NUMBER OF VERTICES (K) = 2 TIMES NO. OF VARIABLES.
      K = 2*NV
C
C    NUMBER OF DISPLACEMENTS ALLOWED.
      NLIM = 5*NV+10
```

```
C                                                                       BXPX3560
C     NUMBER OF CONSECUTIVE TRIALS WITH UNCHANGED FE TO TERMINATE.      BXPX3570
         NCT = NLIM+NV                                                  BXPX3580
         ALPHA = 1.3                                                    BXPX3590
         FK = K                                                         BXPX3600
         FKM = FK-1.                                                    BXPX3610
         BETA = ALPHA+1.                                                BXPX3620
C                                                                       BXPX3630
C     INSURE SEED OF RANDOM NUMBER GENERATOR IS ODD.                    BXPX3640
         IQR = R*1.E7                                                   BXPX3650
         IF (MOD(IQR,2).EQ.0) IQR=IQR+101                               BXPX3660
C                                                                       BXPX3670
C                        SET UP INITIAL VERTICES                        BXPX3680
         FUN(1) = FE(V(1,1))                                            BXPX3690
         YMN = FUN(1)                                                   BXPX3700
      6  FI = 1.                                                        BXPX3710
         FUNOLD = FUN(1)                                                BXPX3720
C                                                                       BXPX3730
         DO 15 I=2,K                                                    BXPX3740
         FI = FI+1.                                                     BXPX3750
         LIMT = 0                                                       BXPX3760
      7  LIMT = LIMT+1                                                  BXPX3770
C                                                                       BXPX3780
C     END CALCULATION IF FEASIBLE CENTROID CANNOT BE FOUND.             BXPX3790
         IF (LIMT.GE.NLIM) GO TO 11                                     BXPX3800
C                                                                       BXPX3810
         DO 8 J=1,NV                                                    BXPX3820
C                                                                       BXPX3830
C     RANDOM NUMBER GENERATOR  (RANDU)                                  BXPX3840
         IQR = IQR*65539                                                BXPX3850
         IF (IQR.LT.0) IQR = IQR+2147483647+1                           BXPX3860
         RQX = IQR                                                      BXPX3870
         RQX = RQX*.4656613E-9                                          BXPX3880
         V(J,I) = BL(J)+RQX*(BU(J)-BL(J))                               BXPX3890
         IF (IP.EQ.1) V(J,I)=AINT(V(J,I)+.5)                            BXPX3900
      8  CONTINUE                                                       BXPX3910
C                                                                       BXPX3920
         DO 10 L=1,NLIM                                                 BXPX3930
         NCE = NCE+1                                                    BXPX3940
         IF (KE(V(1,I)).EQ.0) GO TO 13                                  BXPX3950
C                                                                       BXPX3960
         DO 9 J=1,NV                                                    BXPX3970
         VT = .5*(V(J,I)+CEN(J))                                        BXPX3980
         IF (IP.EQ.1) VT = AINT(VT+.5)                                  BXPX3990
         V(J,I) = VT                                                    BXPX4000
      9  CONTINUE                                                       BXPX4010
C                                                                       BXPX4020
     10  CONTINUE                                                       BXPX4030
```

```
C                                                                       BXPX4040
   11 IF (NPR.LE.0) GO TO 12                                            BXPX4050
      WRITE (6,51) I                                                    BXPX4060
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,I,FUN,CEN,I)                   BXPX4070
   12 IER = -1                                                          BXPX4080
      GO TO 48                                                          BXPX4090
C                                                                       BXPX4100
   13 DO 14 J=1,NV                                                      BXPX4110
      SUM(J) = SUM(J)+V(J,I)                                            BXPX4120
   14 CEN(J) = SUM(J)/FI                                                BXPX4130
C                                                                       BXPX4140
C    TRY TO ASSURE FEASIBLE CENTROID FOR STARTING.                      BXPX4150
      NCE = NCE+1                                                       BXPX4160
      IF (KE(CEN).NE.0) GO TO 7                                         BXPX4170
      NFE = NFE+1                                                       BXPX4180
      FUN(I) = FE(V(1,I))                                               BXPX4190
   15 CONTINUE                                                          BXPX4200
C                                                                       BXPX4210
C    END OF LOOP SETTING OF INITIAL COMPLEX.                            BXPX4220
      IF (NPR.LE.0) GO TO 17                                            BXPX4230
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,0)                   BXPX4240
C                                                                       BXPX4250
C    FIND THE WORST VERTEX, THE 'J'TH.                                  BXPX4260
      J = 1                                                             BXPX4270
C                                                                       BXPX4280
      DO 16 I=2,K                                                       BXPX4290
      IF (FUN(J).GE.FUN(I)) GO TO 16                                    BXPX4300
      J = I                                                             BXPX4310
   16 CONTINUE                                                          BXPX4320
C                                                                       BXPX4330
C    BASIC LOOP.  ELIMINATE EACH WORST VERTEX IN TURN.  IT MUST BECOME  BXPX4340
C    NO LONGER WORST, NOT MERELY IMPROVED.  FIND NEXT-TO-WORST VERTEX,  BXPX4350
C    THE 'JN'TH ONE.                                                    BXPX4360
   17 JN = 1                                                            BXPX4370
      IF (J.EQ.1) JN = 2                                                BXPX4380
C                                                                       BXPX4390
      DO 18 I=1,K                                                       BXPX4400
      IF (I.EQ.J) GO TO 18                                              BXPX4410
      IF (FUN(JN).GE.FUN(I)) GO TO 18                                   BXPX4420
      JN = I                                                            BXPX4430
   18 CONTINUE                                                          BXPX4440
C                                                                       BXPX4450
C    LIMT = NUMBER OF MOVES DURING THIS TRIAL TOWARD THE CENTROID       BXPX4460
C     DUE TO FUNCTION VALUE.                                            BXPX4470
      LIMT = 1                                                          BXPX4480
C                                                                       BXPX4490
C    COMPUTE CENTROID AND OVER REFLECT WORST VERTEX.                    BXPX4500
C                                                                       BXPX4510
```

```
      DO 19 I=1,NV                                                      BXPX4520
      VT = V(I,J)                                                       BXPX4530
      SUM(I) = SUM(I)-VT                                                BXPX4540
      CEN(I) = SUM(I)/FKM                                               BXPX4550
      VT = BETA*CEN(I)-ALPHA*VT                                         BXPX4560
      IF (IP.EQ.1) VT = AINT(VT+.5)                                     BXPX4570
C                                                                       BXPX4580
C    INSURE THE EXPLICIT CONSTRAINTS ARE OBSERVED.                      BXPX4590
   19 V(I,J) = AMAX1(AMIN1(VT,BU(I)),BL(I))                            BXPX4600
C                                                                       BXPX4610
      NT = NT+1                                                         BXPX4620
C                                                                       BXPX4630
C    CHECK FOR IMPLICIT CONSTRAINT VIOLATION.                           BXPX4640
C                                                                       BXPX4650
   20 DO 25 N=1,NLIM                                                    BXPX4660
      NCE = NCE+1                                                       BXPX4670
      IF (KE(V(1,J)).EQ.0) GO TO 26                                     BXPX4680
C                                                                       BXPX4690
C    EVERY 'KV'TH TIME, OVER-REFLECT THE OFFENDING VERTEX THROUGH THE   BXPX4700
C    BEST VERTEX.                                                       BXPX4710
      IF (MOD(N,KV).NE.0) GO TO 22                                      BXPX4720
      CALL FBV (K,FUN,M)                                                BXPX4730
C                                                                       BXPX4740
      DO 21 I=1,NV                                                      BXPX4750
      VT = BETA*V(I,M)-ALPHA*V(I,J)                                     BXPX4760
      IF (IP.EQ.1) VT = AINT(VT+.5)                                     BXPX4770
   21 V(I,J) = AMAX1(AMIN1(VT,BU(I)),BL(I))                            BXPX4780
C                                                                       BXPX4790
      GO TO 24                                                          BXPX4800
C                                                                       BXPX4810
C    CONSTRAINT VIOLATION:  MOVE NEW POINT TOWARD CENTROID.             BXPX4820
C                                                                       BXPX4830
   22 DO 23 I=1,NV                                                      BXPX4840
      VT = .5*(CEN(I)+V(I,J))                                           BXPX4850
      IF (IP.EQ.1) VT = AINT(VT+.5)                                     BXPX4860
      V(I,J) = VT                                                       BXPX4870
   23 CONTINUE                                                          BXPX4880
C                                                                       BXPX4890
   24 NT = NT+1                                                         BXPX4900
   25 CONTINUE                                                          BXPX4910
C                                                                       BXPX4920
      IER = 1                                                           BXPX4930
C                                                                       BXPX4940
C    CANNOT GET FEASIBLE VERTEX BY MOVING TOWARD CENTROID,               BXPX4950
C    OR BY OVER-REFLECTING THRU THE BEST VERTEX.                        BXPX4960
      IF (NPR.LE.0) GO TO 42                                            BXPX4970
      WRITE (6,52) NT,J                                                 BXPX4980
      CALL BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,J)                   BXPX4990
      GO TO 42                                                          BXPX5000
C                                                                       BXPX5010
C   FEASIBLE VERTEX FOUND, EVALUATE THE OBJECTIVE FUNCTION.             BXPX5020
   26 NFE = NFE+1                                                       BXPX5030
      FUNTRY = FE(V(1,J))                                               BXPX5040
C                                                                       BXPX5050
C   TEST TO SEE IF FUNCTION VALUE HAS NOT CHANGED.                      BXPX5060
      AFO = ABS(FUNTRY-FUNOLD)                                          BXPX5070
      AMX = AMAX1(ABS(EP*FUNOLD),EP)                                    BXPX5080
C                                                                       BXPX5090
C   ACTIVATE THE FOLLOWING TWO STATEMENTS FOR DIAGNOSTIC PURPOSES ONLY. BXPX5100
C     WRITE (6,99) J,AFO,AMX,FUNTRY,FUNOLD,FUN(J),FUN(JN),NTFS,N        BXPX5110
C  99 FORMAT (1X,I3,6E15.7,2I5)                                         BXPX5120
      IF (AFO.GT.AMX) GO TO 27                                          BXPX5130
      NTFS = NTFS+1                                                     BXPX5140
      IF (NTFS.LT.NCT) GO TO 28                                         BXPX5150
      IER = 0                                                           BXPX5160
      IF (NPP.LE.0) GO TO 42                                            BXPX5170
      WRITE (6,53) K                                                    BXPX5180
      GO TO 42                                                          BXPX5190
   27 NTFS = 0                                                          BXPX5200
C                                                                       BXPX5210
C   IS THE NEW VERTEX NO LONGER WORST?                                  BXPX5220
   28 IF (FUNTRY.LT.FUN(JN)) GO TO 34                                   BXPX5230
C                                                                       BXPX5240
C   TRIAL VERTEX IS STILL WORST; ADJUST TOWARD CENTROID.                BXPX5250
C   EVERY 'KV'TH TIME, OVER-REFLECT THE OFFENDING VERTEX THROUGH THE    BXPX5260
C   BEST VERTEX.                                                        BXPX5270
      LIMT = LIMT+1                                                     BXPX5280
      IF (MOD(LIMT,KV).NE.0) GO TO 30                                   BXPX5290
      CALL FBV (K,FUN,M)                                                BXPX5300
C                                                                       BXPX5310
      DO 29 I=1,NV                                                      BXPX5320
      VT = BETA*V(I,M)-ALPHA*V(I,J)                                     BXPX5330
      IF (IP.EQ.1) VT = AINT(VT+.5)                                     BXPX5340
   29 V(I,J) = AMAX1(AMIN1(VT,BU(I)),BL(I))                             BXPX5350
C                                                                       BXPX5360
      GO TO 32                                                          BXPX5370
C                                                                       BXPX5380
   30 DO 31 I=1,NV                                                      BXPX5390
      VT = .5*(CEN(I)+V(I,J))                                           BXPX5400
      IF (IP.EQ.1) VT = AINT(VT+.5)                                     BXPX5410
      V(I,J) = VT                                                       BXPX5420
   31 CONTINUE                                                          BXPX5430
C                                                                       BXPX5440
   32 IF (LIMT.LT.NLIM) GO TO 33                                        BXPX5450
C                                                                       BXPX5460
C   CANNOT MAKE THE 'J'TH VERTEX NO LONGER WORST BY DISPLACING TOWARD   BXPX5470
```

```
C     THE CENTROID OR BY OVER-REFLECTING THRU THE BEST VERTEX.           BXPX5480
         IER = 2                                                        BXPX5490
         IF (NPR.GT.0) WRITE (6,52) NT,J                                BXPX5500
         GO TO 42                                                       BXPX5510
      33 NT = NT+1                                                      BXPX5520
         GO TO 20                                                       BXPX5530
C                                                                       BXPX5540
C     SUCCESS:  WE HAVE A REPLACEMENT FOR VERTEX J.                     BXPX5550
      34 FUN(J) = FUNTRY                                                BXPX5560
         FUNOLD = FUNTRY                                                BXPX5570
         NPT = NPT+1                                                    BXPX5580
C                                                                       BXPX5590
C     EVERY 100'TH PERMISSIBLE TRIAL, RECOMPUTE CENTROID SUMMATION TO   BXPX5600
C     AVOID CREEPING ERROR.                                             BXPX5610
         IF (MOD(NPT,100).NE.0) GO TO 37                                BXPX5620
C                                                                       BXPX5630
         DO 36 I=1,NV                                                   BXPX5640
         SUM(I) = 0.                                                    BXPX5650
C                                                                       BXPX5660
         DO 35 N=1,K                                                    BXPX5670
      35 SUM(I) = SUM(I)+V(I,N)                                         BXPX5680
C                                                                       BXPX5690
         CEN(I) = SUM(I)/FK                                             BXPX5700
      36 CONTINUE                                                       BXPX5710
C                                                                       BXPX5720
         LC = 0                                                         BXPX5730
         GO TO 39                                                       BXPX5740
C                                                                       BXPX5750
      37 DO 38 I=1,NV                                                   BXPX5760
      38 SUM(I) = SUM(I)+V(I,J)                                         BXPX5770
C                                                                       BXPX5780
         LC = J                                                         BXPX5790
C                                                                       BXPX5800
      39 IF (NPR.LE.0) GO TO 40                                         BXPX5810
         IF (MOD(NPT,NPR).NE.0) GO TO 40                                BXPX5820
C                                                                       BXPX5830
         CALL BOLT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,CEN,LC)               BXPX5840
C                                                                       BXPX5850
C     HAS THE MAX. NUMBER OF TRIALS BEEN REACHED WITHOUT CONVERGENCE?   BXPX5860
C     IF NOT, GO TO NEW TRIAL.                                          BXPX5870
      40 IF (NT.GE.NTA) GO TO 41                                        BXPX5880
C                                                                       BXPX5890
C     NEXT-TO-WORST VERTEX NOW BECOMES WORST.                           BXPX5900
         J = JN                                                         BXPX5910
         GO TO 17                                                       BXPX5920
      41 IER = 3                                                        BXPX5930
         IF (NPR.GT.0) WRITE (6,54)                                     BXPX5940
C                                                                       BXPX5950
```

```
C     COLLECTOR POINT FOR ALL ENDINGS.                                 BXPX5960
C      1)  CANNOT DEVELOP FEASIBLE VERTEX.            IER = 1          BXPX5970
C      2)  CANNOT DEVELOP A NO-LONGER-WORST VERTEX.   IER = 2          BXPX5980
C      3)  FUNCTION VALUE UNCHANGED FOR K TRIALS.     IER = 0          BXPX5990
C      4)  LIMIT ON TRIALS REACHED.                   IER = 3          BXPX6000
C      5)  CANNOT FIND FEASIBLE VERTEX AT START.      IER = -1         BXPX6010
   42 CONTINUE                                                         BXPX6020
C                                                                      BXPX6030
C     FIND BEST VERTEX.                                                BXPX6040
      CALL FBV (K,FUN,M)                                               BXPX6050
      IF (IER.GE.3) GO TO 44                                           BXPX6060
C                                                                      BXPX6070
C     RESTART IF THIS SOLUTION IS SIGNIFICANTLY BETTER THAN THE PREVIOUS, BXPX6080
C     OR IF THIS IS THE FIRST TRY.                                     BXPX6090
      IF (NPR.LE.0) GO TO 43                                           BXPX6100
      WRITE (6,55) (M,YMN,FUN(M))                                      BXPX6110
   43 IF (FUN(M).GE.YMN) GO TO 47                                      BXPX6120
      IF (ABS(FUN(M)-YMN).LE.AMAX1(EP,EP*YMN)) GO TO 47                BXPX6130
C                                                                      BXPX6140
C     GIVE IT ANOTHER TRY UNLESS LIMIT ON TRIALS REACHED.              BXPX6150
   44 YMN = FUN(M)                                                     BXPX6160
      FUN(1) = FUN(M)                                                  BXPX6170
C                                                                      BXPX6180
      DO 45 I=1,NV                                                     BXPX6190
      CEN(I) = V(I,M)                                                  BXPX6200
      SUM(I) = V(I,M)                                                  BXPX6210
   45 V(I,1) = V(I,M)                                                  BXPX6220
C                                                                      BXPX6230
      DO 46 I=1,NVT                                                    BXPX6240
   46 XS(I) = V(I,M)                                                   BXPX6250
C                                                                      BXPX6260
      IF (IER.LT.3) GO TO 6                                            BXPX6270
   47 IF (NPR.LE.0) GO TO 48                                           BXPX6280
      CALL BOLT (NT,NPT,NFE,NCE,NV,NVT,V,K,FUN,V(1,M),-1)              BXPX6290
      WRITE (6,56) FUN(M)                                              BXPX6300
   48 RETURN                                                           BXPX6310
C                                                                      BXPX6320
   49 FORMAT (50H0INDEX AND DIRECTION OF OUTLYING VARIABLE AT START15)  BXPX6330
   50 FORMAT (50H0IMPLICIT CONSTRAINT VIOLATED AT START. DEAD END. )   BXPX6340
   51 FORMAT ('0CANNOT FIND FEASIBLE',I4,'TH VERTEX OR CENTROID AT STARTBXPX6350
     1.')                                                              BXPX6360
   52 FORMAT (10H0AT TRIAL I4,54H CANNOT FIND FEASIBLE VERTEX WHICH IS NBXPX6370
     1O LONGER WORST,I4,15X,'RESTART FROM BEST VERTEX.')               BXPX6380
   53 FORMAT (40H0FUNCTION HAS BEEN ALMOST UNCHANGED FOR I5,7H TRIALS)  BXPX6390
   54 FORMAT (27H0LIMIT ON TRIALS EXCEEDED. )                          BXPX6400
   55 FORMAT ('0BEST VERTEX IS NO.',I3,'  OLD MIN WAS ',E15.7,         BXPX6410
     1 '  NEW MIN IS ',E15.7)                                          BXPX6420
   56 FORMAT ('0MIN OBJECTIVE FUNCTION IS ',E15.7)                      BXPX6430
```

```
      END                                                               BXPX6440
      SUBROUTINE FBV (K,FUN,M)                                          BXPX6450
      DIMENSION FUN(50)                                                 BXPX6460
      M = 1                                                             BXPX6470
C                                                                       BXPX6480
      DO 1 I=2,K                                                        BXPX6490
      IF (FUN(M).LE.FUN(I)) GO TO 1                                     BXPX6500
      M = I                                                             BXPX6510
    1 CONTINUE                                                          BXPX6520
C                                                                       BXPX6530
      RETURN                                                            BXPX6540
      END                                                               BXPX6550
      SUBROUTINE BOUT (NT,NPT,NFE,NCE,NV,NVT,V,K,FN,C,IK)               BXPX6560
      DIMENSION V(50,50), FN(50), C(25)                                 BXPX6570
      WRITE (6,4) NT,NPT,NFE,NCE                                        BXPX6580
C                                                                       BXPX6590
      DO 1 I=1,K                                                        BXPX6600
      WRITE (6,5) FN(I),(V(J,I),J=1,NV)                                 BXPX6610
      IF (NVT.LE.NV) GO TO 1                                            BXPX6620
      NVP = NV+1                                                        BXPX6630
      WRITE (6,6) (V(J,I),J=NVP,NVT)                                    BXPX6640
    1 CONTINUE                                                          BXPX6650
C                                                                       BXPX6660
      IF (IK.NE.0) GO TO 2                                              BXPX6670
C                                                                       BXPX6680
      WRITE (6,7) (C(I),I=1,NV)                                         BXPX6690
      RETURN                                                            BXPX6700
    2 IF (IK.GE.0) GO TO 3                                              BXPX6710
      WRITE (6,8) (C(I),I=1,NV)                                         BXPX6720
      RETURN                                                            BXPX6730
    3 WRITE (6,9) IK,(C(I),I=1,NV)                                      BXPX6740
      RETURN                                                            BXPX6750
C                                                                       BXPX6760
    4 FORMAT ('0NO. TOTAL TRIALS = ',I5,4X,'NO. FEASIBLE TRIALS = ',    BXPX6770
     1I5,4X,'NO. FUNCTION EVALUATIONS = ',I5,4X,'NO. CONSTRAINT EVALUATIBXPX6780
     2ONS = ',I5/'0      FUNCTION VALUE',6X,'INDEPENDENT VARIABLES/DEPENDBXPX6790
     3ENT OR IMPLICIT CONSTRAINTS')                                     BXPX6800
    5 FORMAT (1H ,E18.7,2X,7E14.7/(21X,7E14.7))                         BXPX6810
    6 FORMAT (21X,7E14.7)                                               BXPX6820
    7 FORMAT (10H0CENTROID 11X,7E14.7/(21X,7E14.7))                     BXPX6830
    8 FORMAT ('0  BEST VERTEX',7X,7E14.7/(21X,7E14.7))                  BXPX6840
    9 FORMAT ('0CENTROID LESS VX',I2,2X,7E14.7/(21X,7E14.7))            BXPX6850
      END                                                               BXPX6860
C     /* CARD GOES HERE
//GO.SYSIN  DD  *
C
C *** THE FOLLOWING CARDS ARE THE DATA DECK FOR PROBLEM III-B.
C *** TACH FEEDBACK COMPENSATION.
```

```
C
1
    1                   1 1000     1
  0.0        0.0002       0.22
  0.7        31.628       1000.
 -0.0349
 -0.0340
 -0.0350
04
BLK  12=C1C2              1000.                  10.
BLK  22=C203              1.
BLK  31=C201              -1.
BLK  41=C301              -1.
```

LIST OF REFERENCES

1. Benchmark Papers in Electrical Engineering and Computer Science, v. K, edited by Director, S. W., Dowden, Hutchinson & Ross, Inc., 1973.

2. McDaniel, W. L. Jr. and Mitchell, J. R., "An Innovative Approach to Compensator Design", NASA CR-2248, May 1973.

3. Office of Naval Research CR 215-238-1, Optimal Linear Control (Formulation to meet Conventional Design Specs.) by G. L. Hartman, C. A. Harvey and C. E. Muller, 29 March 1976.

4. Cantalapiedra, j., Low Order Modles for Dynamic Systems, M.S. Thesis, Naval Postgraduate School, Monterey, 1972.

5. MacNamara, M. A. S., A New Optimization Method Applied to Autopilot Design, M. S. Thesis, Naval Postgraduate School, Monterey, 1975.

6. Box M. J., "A New Method of Constrained Optimization and a Comparison With Other Methods", Computer Journal, 8 April 1965, p. 45 - 52.

7. Fitzgerald, A. B. and Kingsley, C. Jr., Electronic Machinery, p. 186, Fig. 4-22, McGraw Hill, New York, 1961.

8. Thaler, G. J., Design of Feedback Systems, Dowden, Hutchinson & Ross, Inc., Stroudburg, Pennsylvania, 1973.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Naval Postgraduate School Monterey, California 93940	2
4.	Professor George J. Thaler, Code 52TR Naval Postgraduate School Monterey, California 93940	5
5.	Lt. Larry P. Vines 119 Indian Lane Oak Ridge, Tennessee 37830	3
6.	Dr. Jerrel R. Mitchell Department of Electrical Engineering Mississippi State University Mississippi State, Mississippi 39762	1
7.	Mr. Jay Cameron IBM Monterey and Cottle Roads San Jose, California 95114	1

8. Mr. Peter Gramata 1
IBM
Monterey and Cottle Roads
San Jose, California 95114

9. Mr. Robert McIntosh 1
IBM
Monterey and Cottle Roads
San Jose, California 95114

10. Mr. Martin Dost 1
IBM
Monterey and Cottle Roads
San Jose, California 95114

11. Mr. Ron Palmer 1
IBM
Monterey and Cottle Roads
San Jose, California 95114

12. Mr. W. M. Syn 1
IBM
Monterey and Cottle Roads
San Jose, California 95114

13. Professor A. G. J. MacFarlane 1
Dept. of Electrical Engineering
University of Manchester
Inst. of Science and Technology
Manchester England 067609

14. Dr. Bui-Tien Rung 1
Dept. of Applied Sciences
Universite du Quebec a Chicoutimi
930 est, rue Jacques-Cartier
Chicoutimi, Quebec
G7H 2B1